63-3-4

JANUARY 1963

DEPARTMENT OF ENGINEERING

annual summary report of investigations in digital technology research

JUNE 1, 1961 - MAY 31, 1962

PROJECT STAFF

REPORT NO. 62-64

UNIVERSITY OF CALIFORNIA, LOS ANGELES

Report No. 62-64
January 1963

ANNUAL SUMMARY REPORT OF INVESTIGATION IN DIGITAL TECHNOLOGY RESEARCH

June 1, 1961 - May 31, 1962

M. Aoki
B. Bussell
G. Estrin
C.T. Leondes

Co-Principal Investigators

J. Bibb
A. Borsei
A. Fraenkel
R. Fuller
J. Marcus
D. Martin
E. Russell
P. Schaeffer
R. Turn
C. Viswanathan

DEPARTMENT OF ENGINEERING
UNIVERSITY OF CALIFORNIA
LOS ANGELES 24, CALIFORNIA

FOREWORD

The research described in this report, *Annual Summary Report of Investigations in Digital Technology Research; June 1, 1961 - May 31, 1962,* by the Project Staff, was carried out under the technical direction of M. Aoki, B. Bussell, G. Estrin, and C.T. Leondes and is part of the continuing program in Digital Technology Research.

This project is conducted under the sponsorship of the Department of the Navy, Office of Naval Research. Submitted in partial fulfillment of Contract Number Nonr-23(52).

TABLE OF CONTENTS

CHAPTER I - COMPUTER ORGANIZATIONS

I-1 Organization and Command Structure for Content-Addressable Memory Systems

1.1.1 Introduction

This report describes some methods for parallel processing of data through use of a content-addressable memory (CAM). Processing is parallel in the sense that all words in CAM or any designated subset of these may be processed in a single operation. Allowable data processing operations and allowable arguments for these are dependent on the logical and physical properties of CAM. The significance of these data processing methods will be evaluated relative to both hardware requirements and execution time.

The objectives of this report are:

1) Define the class of data processing operations which are feasible in CAM and which may make use of the parallelism of CAM.
2) Describe some variations in logical organization of CAM. Describe the data processing operations which are made more or less effective by each variation. For a selected CAM organization, present an extendable command set and a control organization to permit mechanization of the set.
3) Define problem characteristics which generally guarantee that solution time will be decreased by use of CAM.

The CAM memory system was initially proposed by Slade, et al[1] to allow retrieval of stored data by reference to content of a cell rather than by its physical location. Cell locations or contents are retrieved, if a specified portion of their content equals a key word. The required equality search is executed in parallel for all cells.

In conventional systems, data is stored in a random access memory as a one-dimensional array. For data not initially in this form a single index mark may be assigned to each memory element according to a "memory mapping function" which is formed on reference properties and assumes positive integral values. For example a three-dimensional array with indices

$i = 0(1)\ (N-1)$, $j = 0(1)\ (N-1)$, $k = 0(1)\ (N-1)$ may be converted to a one-dimensional array with index $l = 0(1)\ (N^3 - 1)$ by the mapping function

$$l = i + Nj + N^2k$$

Data having given reference properties are assigned a storage location determined from the mapping function evaluated for these properties. Data may be retrieved by location addressing with knowledge of the reference properties, the mapping function and the correspondence of mapping function values to storage locations.

Location addressing becomes cumbersome and content-addressing efficient under conditions described below:

1) Data is to be addressed by several sets of reference properties. If location addressing is used, several mapping functions must be defined and the data item or reference to it must be stored with a multiplicity equal to the number of possible sets of reference properties. If this number is unknown, location addressing is unusable without observation of every piece of stored data with possible reordering and relocation. An example would be a table of function values which is referenced by one or more function values as well as by one or more arguments.

2) Data elements are sparse relative to values of the reference property. Location addressing is often forced to assign cells for which no data are stored. An example would be a sparse matrix, where each element is stored according to a mapping function on its matrix indices with the resulting assignment of a separate cell to each null element.

3) Data become dynamically disordered in memory during processing. If it becomes necessary during processing to refer to data by some property, reordering and relocation will be necessary using location addressing.

Other uses for CAM will become evident as its properties are detailed. It is evident that the memory mapping function need not be single valued for CAM;

hence that multiple-membered sets of CAM cells may be defined and addressed. In the next section some possible set operations for CAM are considered.

1.1.2 Set Operations in CAM

Each word cell in CAM may be defined by content of the "extended cell" as a member of arbitrary sets, S. The extended cell includes bit positions in the memory matrix and in any external but associated storage elements. Any data processing operation in CAM may be considered as a set operation executed over sets, S. A useful list of set operations executable in CAM is presented below. The particular command structure presented for the CAM system described later in this chapter synthesizes many of these operations.

It is assumed that CAM exists as a subsystem for a controlling computer. The computer can present some "key" or reference word to CAM, generate arbitrary functions of this key, present some "tag" word consisting of "1's", "0's", and "don't care's" to define set marks, call for data stored in CAM, present data to CAM for storage and execute various tests defined below.

1) Mark a set S. Mark the set of CAM cells having some element of their Boolean character in common. A mark exists as unique states for specified bit positions of the extended cell. Boolean character of a cell is defined as some Boolean function of cell contents and of contents of a reference or "key" word. Cells of a given Boolean character may be identified and marked if a set of primitive Boolean operators is mechanized at each word cell in CAM.

The following operations may be executed over marked sets.

2) Mark the sum or union of sets S_1 and S_2 in CAM. Mark the set of CAM cells which belong to at least one of the sets S_1 and S_2.
3) Mark the intersection of sets S_1 and S_2 in CAM. Mark the set of CAM cells belonging to both S_1 and S_2.
4) Mark the difference of sets, S_1 minus S_2. Mark the set of CAM cells contained in S_1 but not contained in S_2.

5) Replace specified bits for cells in some set, S. For example:

   a) Store data into a single cell having a given mark. The mark may be changed in this process.
   b) Store separate data items into each cell having a given mark.
   c) Replace specified data or mark bits with defined bits for each cell having a given mark. For example, void a set by erasing a set mark.
   d) Store data at some externally specified location.

6) Retrieve set members stored in CAM. For example:

   a) Read contents of a single cell having a given mark.
   b) Read contents of all cells having a given mark.
   c) Read contents of a cell at some externally specified location.

7) Test for the number of members in a given set.

   a) Test if a set is void. Does any CAM cell have a given mark?
   b) Test if a set has more than a single member. Does more than one CAM cell have a given mark?
   c) Test if a set equals the set of all CAM cells. Do all CAM cells have a given mark?
   d) Test if a set lacks more than a single member to equal the set of all CAM cells. Does more than one CAM cell lack a single mark?

If contents of CAM cells are interpreted as numeric quantities, the following set operations may be defined and executed:

8) Locate the member or members of some input set having the largest (smallest) signed or absolute value in some specified field of bit positions. Members of the input set having this property are marked as an output set.

9) Locate the member or members of some input set having a value for some specified bit field which is $>, \geq, <, \leq, =, \neq$ a key. Members of the input set having this property are marked as an output set.

10) Execute arithmetic operations for sets of number pairs where each pair included in the set may be defined in the following ways:

   a) One member of a pair is an externally derived key common to all pairs; the second member is stored in a bit field for each CAM cell.

   b) Pairs are stored in two bit fields within a single CAM cell.

   c) A bit field is specified for each cell in each of two sets. To each field in one set there corresponds a field in the other set and conversely. Pairs are stored in corresponding fields.

Examples of arithmetic operations are:

   a) Form the sum of a pair.

   b) Form the product of a pair.

The CAM system requirements for mechanizing these set operations are discussed in the following section.

1.1.3 Organization of CAM Systems

A word cell in CAM may be referenced by some address operation (a Boolean function of cell contents and of bits in some key word) rather than (or in addition to) reference by cell location. One or more operators are mechanized at each CAM cell and may be evaluated simultaneously for all cells. Contents of cells are not destroyed in evaluation of an address operator. Results of evaluation are stored in a detector.

Typical address operators are the compare operators ($>, \geq, <, \leq, =, \neq$) operating on signed numbers or magnitudes. One or more compare operators could be mechanized at each cell and would compare content of that cell to an externally presented key in a sense defined by the operator. The operator has the value 1 if the defined comparison is satisfied, and 0 if it is not.

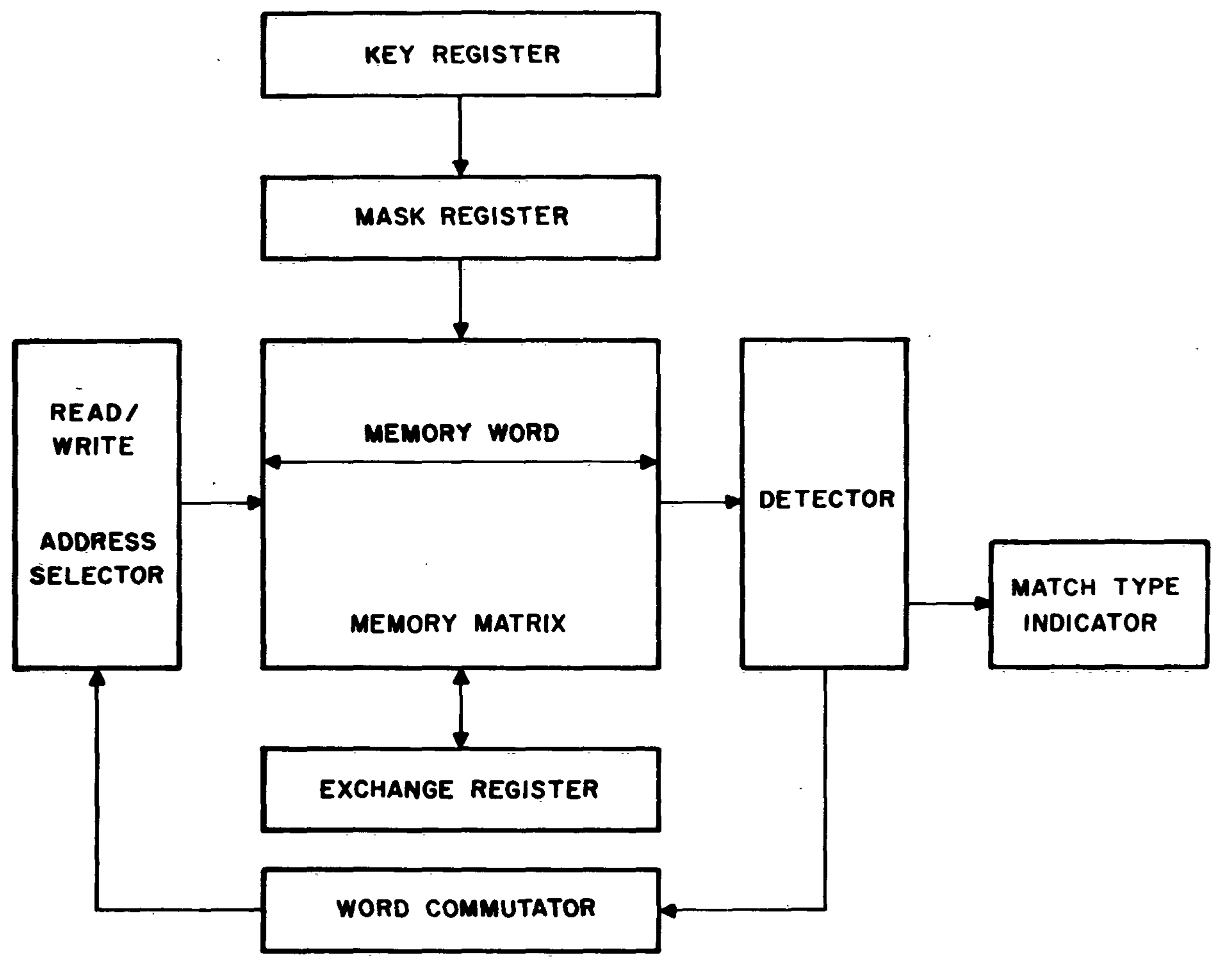
Functional blocks for a possible CAM system are illustrated in Figure 1.1.

The mask register, if present, is 1 in those bit positions for which contents of the key register and memory matrix serve as arguments for an address operator and is 0 elsewhere. Bit positions for which the mask register is 0 are said to be "masked". Contents of the mask register and of the key register may be arbitrarily specified under program control.

The exchange register required to write data into memory and read data from memory may be a unique register or this function may be served by the key register.

The detector may contain a single storage element which is set true when an address operator is satisfied for some word within memory. The equality operator would generally be mechanized for this simple type of detector. Addresses for words satisfying the address operator are not available. Match type indication (i.e., indication of a multiplicity of words satisfying the address operator) may or may not be available. Alternatively the detector may be a plane containing a mark storage element for each word cell in memory. Any address operator may then be mechanized. However, in this instance, equality comparison with summation of successive marked sets in the detector plane is a primitive which, repeatedly applied, allows synthesis of any address operator. An equality comparison is defined to leave detector elements true for word cells which match a masked key.

Address operators and detector elements may be desired with high multiplicity; hence it is desirable that their logical properties be formulated so as to imply inexpensive mechanization. The logical network which mechanizes each address operator is assumed to be quiescently in its false state. The true state is entered transiently if the address operator is satisfied during an evaluation. The state of a detector element may be altered by a true address operator and is not altered by a false operator.



BLOCK DIAGRAM FOR CAM SYSTEM
FIGURE 1.1

An equality comparison may be mechanized through use of either the equality or the inequality address operator. If the equality operator is used, detector elements are initially reset and are then set by a true operator. If the inequality operator is used, detector elements are initially set and are then reset by a true operator. Successive evaluations of the equality operator, with no intervening reset of the detector, leave a detector element true if contents of its word cell match at least one masked key. The resulting set is the sum of sets matching individual keys. Successive evaluations of the inequality operator, with no intervening set of the detector, leave a detector element true if contents of its word cell match all masked keys. The resulting set is the intersection of sets matching individual keys. For logical properties described above, the equality operator must be evaluated parallel-by-bit (i.e. for all the bits in a masked key). The inequality operator may be evaluated either serial- or parallel-by-bit.

To read from or write into a cell marked by a detector plane, some link (Figure 1.1) is required between the detector and the read-write address selector. If many cells are marked by the detector, a word commutator is required in this link to select a single cell, or to sequentially select all cells, for reading or writing.

If linear rather than coincident address selection is employed, it is further possible to replace, in parallel, specified bits in detector-marked cells by defined bits. In particular, control bits may be stored within the memory matrix. Control bits mark sets as do elements of the detector plane but do not link the address selector. They may be cleared to "0" throughout CAM and may be set to "1" or "0" in detector-marked cells.

Having at least one control bit in addition to a detector element for each CAM cell, (the bit may be stored internal or external to the memory matrix) and using either the equality or inequality address operator, it is possible to mark the complement set of the detector-marked set. The control bit is written to "1" in detector-marked cells and subsequently interrogated for "0". It is possible to mark the sum of arbitrary detector-marked sets by resetting the control bit for all cells and subsequently writing it to "1" for each detector-marked set in turn. It is possible to mark the intersection of detector-marked sets by writing the control bit to "1" for all cells, subsequently resetting it for each detector marked set and finally complementing it. It is possible to mark the difference of sets, S_1 minus S_2, by complementing S_2 and marking the intersection of this set with S_1 by methods illustrated above.

If a CAM system contains a detector plane, a word commutator, and is linearly selected such that parallel replacement may occur in detector marked cells, set members may be altered and retrieved in all senses defined in Section 1.1.2. If this CAM system is further mechanized to test for the presence of the void or of multiple membered sets in the detector plane, all algorithms listed in Tables I, II and III and described in Section 1.1.6 are executable.

Cryogenic CAM systems having many of these capabilities have been described by Seeber, Davies and others.[5, 6, 7, 8, 10, 11] The proposed system differs from previously described systems in that parallel replacement is generalized to all bit positions in memory. Control bits are stored in the memory matrix and may be evaluated by the address operator. For previously described CAMs, parallel replacement was possible only in certain specially wired bit positions designated as mark or control bits which were stored external to the memory matrix. When these capabilities are provided, the ability to identify and transform sets is significantly extended.

For CAM systems which are coincidence selected (rather than linearly selected) it is not possible to replace bits in parallel over many cells. The arithmetic algorithms (additions, multiplications and format conversions) discussed in Section 1.1.6 are not executable. Control bits must be stored external to the memory matrix.

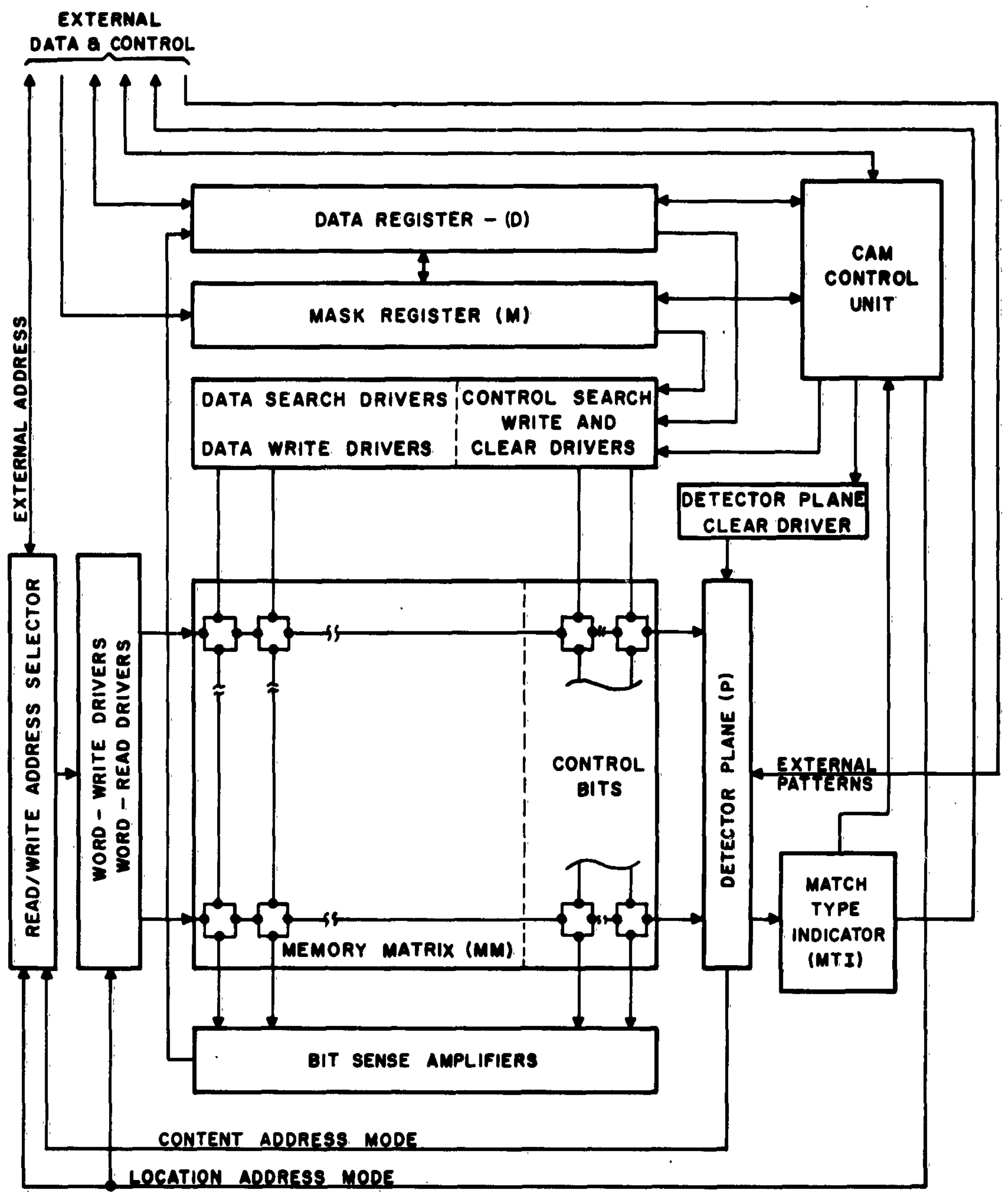
For CAM systems having a detector plane linked to a read-write address selector but no word commutator, the detector-marked set must be reduced to a single member in order to access this member for reading or writing. Conventional coordinate addressing may also be possible. It is probable that tests for void and multiple-membered detector-marked sets would be incorporated into this system. A coincidence-selected CAM system having essentially this capability and using the inequality address operator has been described by Petersen and co-workers.[4]

A CAM system having a single detector element (rather than a plane) and the equality address operator has been proposed by Slade and co-workers.[1, 2] Data are read from this memory by executing a serial-by-bit match test. Cells are location-addressed for writing.

1.1.4 Proposed CAM System

It is now desired to particularize the description of CAM to the extent necessary for discussion of execution sequences and timing for specific algorithms. A block diagram for the proposed CAM system is shown in Figure 1.2. The system has the following properties:

1) The equality comparison (using either the equality or the inequality address operator) is mechanized.
2) Memory elements are nondestructively interrogated and outputs are logically combined for each word to form the address operator. Detector elements (P) are initially cleared to the matched state and are then set to the mismatched state by a true address operator. Masked bits (M=0) are not interrogated.
3) A single data register (D) serves the function of exchange and key registers.
4) The match type indicator (MTI) generates one of three signals indicating the multiplicity of matching words. These are:
    a) No matching word.
    b) Single matching word.
    c) Multiple matching words.
5) A single uniquely matched word may be selected for reading or writing. Reading is nondestructive. Selection may be on



BLOCK DIAGRAM FOR PROPOSED CAM SYSTEM

FIGURE 1.2

the basis of an externally supplied address as well as on the basis of detector-plane contents.

6) The address selector is linked to the detector through a word commutator which allows multiple words to be sequentially selected for reading or writing without repeating the search for each word processed.

7) A single bit position for a set of nonuniquely matched words may be altered to either the "1" or "0" state. This bit may be any selected bit of the word and is altered in parallel for all words.

8) A group of control bits is defined for each word in CAM. Each bit designates the word as a member of some set. Control bits are stored within the memory matrix. They are searched and written in configurations specified by a tag contained within commands which use them. The tag specifies each control bit to be "1", "0" or masked.

The inequality comparison is chosen because it allows simple summation of outputs from memory elements to generate the required Exclusive Or function. Read-out may be dynamic and pulses need not be in time coincidence. Interrogation of successive bits can be delayed sufficiently to reduce objectional noise buildup. Masking is simply accomplished by inhibited interrogation of masked bit positions.

The ability to sequentially select members of a matching set allows simple loading of the memory in instances where locations of vacant cells are unknown to the program. The vacant set is selected by searching on appropriate control bits and words are sequentially written into this set. Circuitry required to accomplish this task is directly usable in the task of sequential reading members of some selected set.

There are instances in which it is desired to address data or vacant cells by location rather than content. It is possible to assign some unique "serial number" to some field of each cell and to perform "location addressing" by searching for this serial number. Alternatively memory addressing circuitry can accept an externally supplied address as well as addresses determined from detector plane contents. The latter technique is employed (without precluding use of the former) when it is more conservative of time and of bit positions in memory.

The proposed memory allows parallel replacement of a single bit in each detector marked word. Control bits may thus be stored in the memory matrix. The algorithms to be presented demonstrate that the extension of this property to all bits is a useful adjunct to the simultaneous search property. In the usual case it is only desired to alter a single bit of each selected set and circuit considerations may limit simultaneous alteration to a single bit. This property requires the memory to be linearly selected.

It may be desirable to store some externally presented pattern of "1' s" and "0' s" into the detector plane and thus select a multi-membered set of CAM cells. As an example, this pattern may be derived from some character recognition device and it may be desired to correlate it in some sense with CAM contents. Having the parallel replacement property, direct parallel entry to the detector plane also allows CAM to be loaded parallel-by-word, serial-by-bit.

It is desirable to distinguish control bits conceptually from data even though both are stored in the memory matrix. Some of the reasons for making this distinction are:

1) Control bits may be cleared to "0" throughout memory, thus erasing set inclusion statements prior to establishing new statements. This operation is not necessarily performed on data bits. Initial reset of the entire memory may be accomplished by sequential writing of null words.

2) The requirement for altering the tag which specifies control bit configurations is often distinct from that for altering the data mask and data key.

1.1.5 Control for Proposed CAM

The proposed CAM is for generality assumed to be a component of a larger computer system. A CAM command is initiated by first loading the data and mask registers and inserting an operation code into a command register. A "START" pulse is then issued to the CAM system. The command is executed asynchronously and a "COMPLETION" pulse is sent to the controlling computer. Execution times for CAM commands are measured from START pulse to COMPLETION pulse.

The CAM control consists of a command sequencer, a search driver selector, a word selector and a set of index registers. The command sequencer controls the microsteps required for execution of a CAM command. The search drive selector provides sequential delay for each bit interrogated during a search command, by passing all masked bits with some lesser delay. The word selector sequentially selects multiple matching words for reading or writing. The scan register serves as a bit index in certain commands which are executed serial-by-bit. Each of these devices is more fully described in the following paragraphs.

The command sequencer (Figure 1.3) contains a command register, state counter, command decoder and an aperiodic pulse generator. The command register stores an operation code and, together with "branch" and "end" signals from the CAM system, controls the sequence of states for the state counter. The command decoder translates contents of the state counter and command register into elementary control signals used throughout the CAM system and in particular to select time delays between successive clock pulses.

The clock generator is triggered by the START pulse and at each following interval by one of a set of delay generators (Ti through Tn in Figure 1.3). Each delay generator has logical inputs (t) and (i). When triggered at the logical input (t), a delay generator transmits a trigger pulse to the clock generator after some preset delay. When triggered at the logical input (i), the delay generator transmits a trigger pulse to the clock generator with no delay. The (t) input is normally triggered by a clock pulse ANDed with a state and an instruction code.

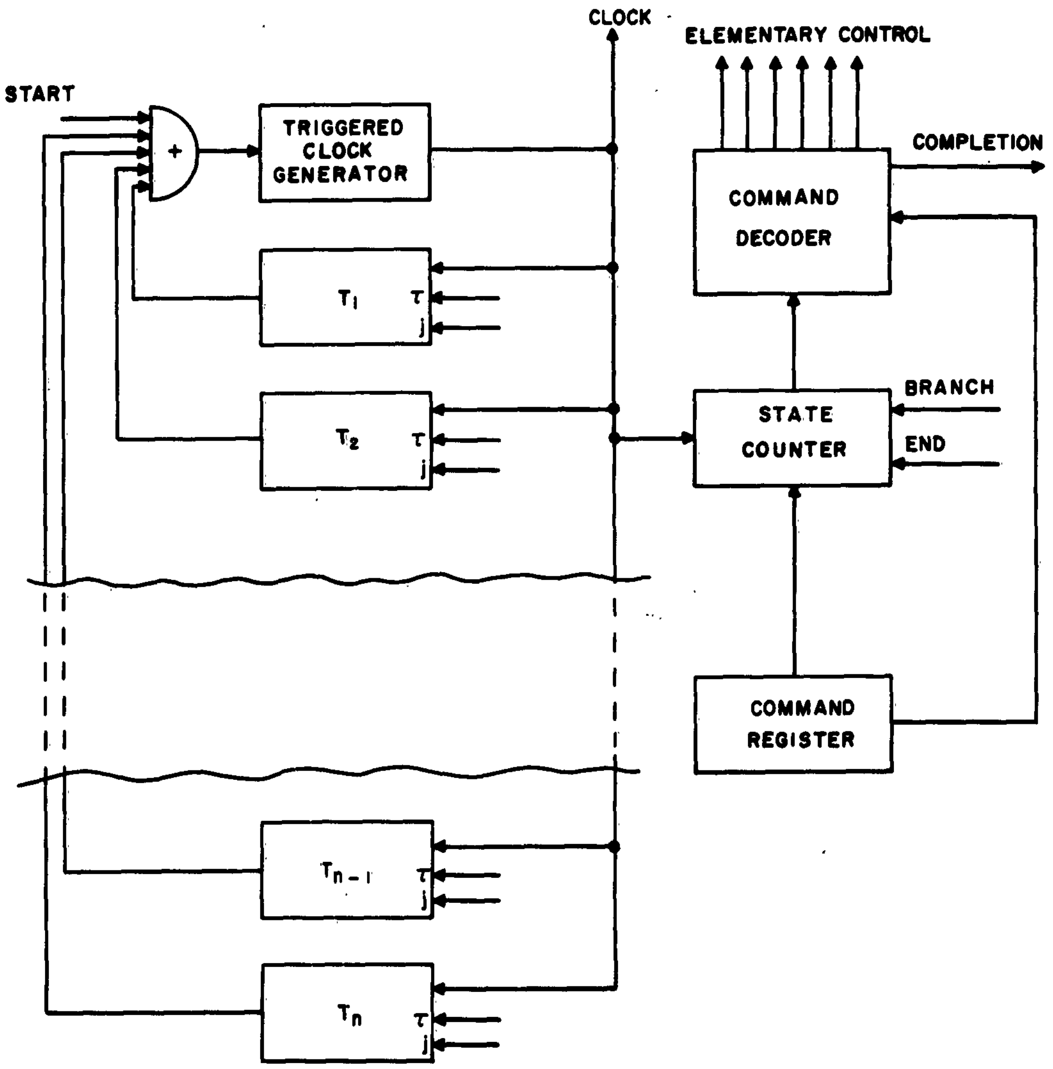
1.1.5 Control for Proposed CAM

The proposed CAM is for generality assumed to be a component of a larger computer system. A CAM command is initiated by first loading the data and mask registers and inserting an operation code into a command register. A "START" pulse is then issued to the CAM system. The command is executed asynchronously and a "COMPLETION" pulse is sent to the controlling computer. Execution times for CAM commands are measured from START pulse to COMPLETION pulse.

The CAM control consists of a command sequencer, a search driver selector, a word selector and a set of index registers. The command sequencer controls the microsteps required for execution of a CAM command. The search drive selector provides sequential delay for each bit interrogated during a search command, by passing all masked bits with some lesser delay. The word selector sequentially selects multiple matching words for reading or writing. The scan register serves as a bit index in certain commands which are executed serial-by-bit. Each of these devices is more fully described in the following paragraphs.

The command sequencer (Figure 1.3) contains a command register, state counter, command decoder and an aperiodic pulse generator. The command register stores an operation code and, together with "branch" and "end" signals from the CAM system, controls the sequence of states for the state counter. The command decoder translates contents of the state counter and command register into elementary control signals used throughout the CAM system and in particular to select time delays between successive clock pulses.

The clock generator is triggered by the START pulse and at each following interval by one of a set of delay generators (Ti through Tn in Figure 1.3). Each delay generator has logical inputs (t) and (i). When triggered at the logical input (t), a delay generator transmits a trigger pulse to the clock generator after some preset delay. When triggered at the logical input (i), the delay generator transmits a trigger pulse to the clock generator with no delay. The (t) input is normally triggered by a clock pulse ANDed with a state and an instruction code.



BLOCK DIAGRAM FOR COMMAND SEQUENCE

FIGURE 1.3

In this mode the delay generator controls the duration of the state which is entered on the clock pulse triggering the delay generator. The (i) input is normally triggered by some completion signal developed within CAM for operations of variable time duration. Both inputs of a delay generator may be wired, allowing normal duration for a state for normal circumstances and premature termination if special conditions are met.

The search drive control provides sequential delay, Tsbd1 for each bit interrogated (M_i = "1") during a search operation, by-passing masked bits (M_i = "0") with some lesser delay, Tsbd2. In many memory mechanizations this delay is of substantial use in reducing noise buildup at matched cells. This circuit is repeated for each bit of the word including control bits. A completion signal is developed at the final stage of the drive control.

The word commutator sequentially selects one of a set of matching words for reading or writing. Interconnection of detector plane storage element, word driver and word commutator element for some "jth" CAM word is shown in Figure 1.4. This circuit is repeated for each word cell in CAM. The signal "C_j" is propagated through a single element in the commutator chain in a time, Tswd. A "select complete" signal is generated to indicate the time at which unique address selection is achieved.

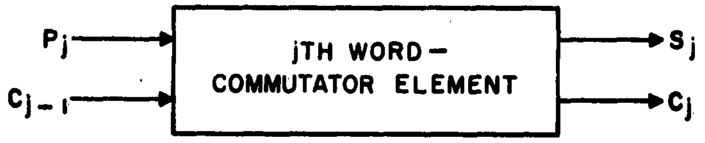


FIGURE 1.4

P_j = 1 denotes that the "jth" detector element is matched.
C_j = 1 denotes that none of words j through h are selected.
S_j = 1 denotes that the "jth" word is selected

$$S_j = P_j \cdot C_{j-1}$$
$$C_j = \overline{S_j} \cdot C_{j-1}$$

The set of index registers is used to form auxiliary masks and keys for compound commands and to count steps in these commands. Each index register consists of an index counter which addresses some bit position in the data or mask registers and of two limit registers. Limit registers are loaded from an index word presented by the controlling computer. A pair of limit registers define a contiguous field of bit positions in CAM. Bit positions are considered consecutively numbered starting at the least significant bit (LSB) of the cell. The address of the LSB of the field is stored in the lower limit registers; the address of the most significant bit (MSB) of the field is stored in the upper limit register. All bit positions having numbers greater than or equal to the LSB but less than or equal to the MSB are considered in the field. The CAM control and the controlling computer can load each index counter from one of its limit registers, compare each counter to its upper or lower limit register and increment or decrement counters and limit registers. Bits in D and M addressed by an index counter may be altered or tested by the CAM control or the controlling computer.

1.1.6 CAM Commands

The controlling computer can issue some number of data processing and control commands to CAM. These commands may be embedded into the conventional command structure of this machine, hence read from memory during an instruction cycle and then executed. Alternatively, the CAM system may be part of a "variable structure" computer as proposed by G. Estrin.[14] Commands would then be sequenced for any given problem by a supervisory control interacting with the variable structure system. Some CAM commands are executed over the entire contents of CAM. Other commands are executed over some "input" set of detector-marked CAM words where the detector is assumed to have been set by some previous search. The detector-marked set at completion of a command is designated as the "output" set.

Commands are designated as "basic commands" or as "compound commands". Each basic command is a useful program step and the set of basic commands serves as building blocks for compound commands. Both basic and compound commands are executed in response to a single instruction from the controlling computer. The CAM control sequences basic commands to execute a compound command.

A set of basic CAM memory commands and basic logical commands are listed in Tables I and II, respectively. A set of compound commands is listed in Table III. Both command sets are, of course, extendable. The search command (SCH) and the maximization command (MAXA) are then discussed in greater detail in the following pages. The notation C(R), defined as "contents of register R" is used in description of commands. The search command (SCH) together with marking commands (CLP, WSB, CLC) allows selection of a set having any given character. The basic write commands (WFW and WSB) allow arbitrary transformation of the selected set. By repeated use of these commands, it is possible to perform arithmetic and logical operations on sets of CAM cells. Operations are usually executed serial-by-bit and parallel-by-word hence execution time is not strongly dependent on the number of words processed.

The controlling computer is assumed to contain an arithmetic unit in which it is possible to execute the usual arithmetic and logical operations on single words at far greater speeds than they can be executed in CAM. A set of compound commands was selected to have some expectation of being executed over large sets of words with a view to determining the feasibility of parallelism sufficient to outperform the conventional arithmetic organ. In a later report consideration will be given to any further properties necessary to permit CAM to perform in a manner equivalent to networks of computers under a common control.

The list of compound commands is merely indicative of some possible choices. It is obvious that any compound command, treated here as a wired command, can be synthesized by repeating basic or lower level compound commands under program control. The choice of wiring a compound command is the common choice of execution time vs. hardware complexity.

In the remainder of this section timing equations for basic and compound commands are described. One basic command (SCH) and one compound command (MAXA) are discussed. For these commands a flow chart is presented which defines elementary events and their sequence of occurrence. Elementary

TABLE I

BASIC MEMORY COMMANDS

Mnemonic Code	Command
1) SCH(Tag, Tagm) (Search)	The detector plane is cleared to the matched state then reset to mismatched state for CAM cells having data contents which do not match the masked key or control bits which do not match the tag masked by Tagm. The input set includes all CAM cells. The output set contains only matching cells.
2) WNW(Tag, Tagm) (Write next word)	Unmasked bits of data register are written into the next sequential cell (according to some defined ordering) of the selected input set. Masked bits for this cell are unaltered. Tag bits masked by Tagm are written into the control field for this cell. The output set equals the input set minus the selected word.
3) WTL(Y, Tag, Tagm) (Write location)	Unmasked bits of the data register are written into the cell at location Y. Masked bits for this cell are unaltered. Tag bits masked by Tagm are written into the control field for this cell. Neither input nor output sets are specified.
4) RNW (Read next word)	Contents of the next sequential cell of the selected input set are read into the data register. The output set equals the input set minus the selected word.
5) RDL(Y) (Read location)	Contents of the cell at location Y are read into the data register. No input or output sets are specified.
6) WSB(Tag, Tagm) (Write single bit)	A single defined bit is written into a defined bit position for all cells of the selected input set. If Tagm is non zero, an unmasked control bit is modified. Otherwise an unmasked data bit is modified. The output set equals the input set.
7) CLC(Tagm) (Clear control bits)	The tag consists only of the mask, Tagm. The control field is cleared to 0 in all bit positions where the mask is 1. No input or output sets are specified.
8) PLP (Place address in Accumulator)	The location of a single matched detector element is converted to an address and stored in the accumulator of the controlling computer.

TABLE II

BASIC LOGICAL COMMANDS

Mnemonic Code	Command
1) LDM (Y) (Load M)	Some fraction of the M register is loaded with C(Y) from memory of the controlling computer.
2) LDD(Y) (Load D)	Some fraction of the D register is loaded with C(Y).
3) STOD(Y) (Store D)	Some fraction of C(D) is stored at address Y.
4) TDMZ (Test D or M for Zero)	The C(D) or the C(M) are tested for zero.
5) LXLR(Y) (Load Index Limits)	Index limit registers are loaded with C(Y).
6) LIX (Load Index)	Index counters are loaded from specified limit registers.
7) LIXSL (Load Index, Step Limit)	Index counters are loaded from specified limit registers. Limit registers are incremented (decremented) by one.
8) TIX (Test Index)	Contents of a specified index counter are tested relative to some limit register.
9) SDMI (Set D and M)	Data and mask registers are set in positions addressed by index counters.
10) TDMI (Test D and M)	Data or mask register is tested in position addressed by some index counter.
11) MTIT (Test MTI)	Match type indicator is tested for zero, one, or greater than one.

TABLE III

COMPOUND COMMANDS

Mnemonic	Command
1) MAXA (Absolute maximum)	Members of some input set which have the maximum absolute value for some field are marked as an output set.
2) MINA (Absolute minimum)	Members of some input set which have the minimum absolute value for some field are marked as an output set.
3) GTK (Greater than key)	Members of some input set which have the numeric value for some field greater than a key are marked as an output set.
4) LTK (Less than key)	Members of some input set which have the numeric value for some field less than a key are marked as an output set.
5) GEK (Greater than or equal to key)	Members of some input set which have the numeric value for some field greater than or equal to a key are marked as an output set.
6) LEK (Less than or equal to key)	Members of some input set which have the numeric value for some field less than or equal to a key are marked as an output set.
7) ADDC (Add constant)	Contents of some field in the D register are added to all cells in some input set. The output set equals cells in which overflow occurs.
8) ADDF (Add fields)	Contents of two fields are added for each cell in some input set. The output set equals cells in which overflow occurs.
9) MPYC (Multiply constant)	The product of some field of C(D) by some field of each cell in some input set is formed in the cell. The output set equals cells in which overflow occurs.
10) MTXMPY (Matrix multiply)	For some row of a matrix A and column of a matrix B, in some input set a term of each element in the product matrix C = AB is formed. The output set equals cells in which overflow occurs.
11) BCDBY (BCD to binary conversion)	Some input set of numbers in binary notation is converted to BCD notation. The output set equals the input set.
12) BYBCD (Binary to BCD conversion)	Some input set of numbers in BCD notation is converted to binary notation. The output set equals the input set.

events are defined assuming that CAM is structured as in the block diagram of Figure 1.2. On each flow chart the state sequence for CAM is indicated and the set of elementary events occurring within a single CAM state is shown.

The controlling computer presents data and mask register contents; an index word and a command code to CAM then issues a START pulse. The CAM system executes the command then issues a COMPLETION pulse. Each command sequence is described for the interval between START and COMPLETION pulse. Some arbitrary command (CMD) is executed in a time, Tcmd, starting and ending in ST 0 with an intermediate state sequence, ST 1 through ST n. Each intermediate state is assigned a time, τcmd1 through τcmdn.

Each state time, τ, is a sum of elementary times. Elementary times are defined on the flow chart for each command and may be common to several commands. Subscripts for elementary times are chosen to be of mnemonic significance. State times may consist of a fixed number of elementary times, hence be of a fixed duration, determined by a delay generator. State times may also consist of a variable number of elementary times, hence be of variable duration, determined by a completion pulse.

Any logical operation on contents of the mask, data or scan register is executed in a single state time, τ_L, and an elementary time, t_L, for all operations and all commands. Any branch condition for control states, implied by a test on contents of the mask, data or index register or on contents of the match type indicator, is evaluated in a state time, τ_L, and an elementary time, t_L, which may overlap other logic states. Each state (including the initial state) is assigned an elementary "advance time", t_{sa}, to allow for transition of the state counter into the following state and for settling of control lines in this state. The state time for any logic state is

$$\tau_L = t_L + t_{sa}$$

Each command has a transition time, $\tau_o = t_{sa}$, from its initial state. A basic command (CMDB) is executed in a time

$$T_{CMDB} = \tau_0 + \sum_{i=1}^{n} \tau_{CMDi}$$

where n is the number of intermediate states. Under some conditions a basic command may be skipped; hence be executed in time, τ_0. A compound command (CMDC) exists as a series of basic commands (CMDB1 through CMDBN) with intervening logic states. A compound command is executed in a time

$$T_{CMDC} = \tau_0 + \alpha_1 \sum_{i=1}^{n_1} \tau_{CMDB1i} + \cdot \cdot \cdot + \alpha_n \sum_{i=1}^{n_N} \tau_{CMDNi} + \ell \tau_L$$

where n_1 through n_N and α_1 through α_n are respectively the number of intermediate states in and the number of iterations of basic commands CMDB1 through CMDBN and ℓ is the number of logic states in the compound command.

Search Command -- SCH(Tag)

The SCH command searches the memory matrix for words which match C(D) in bit positions where C(M) are 1. No match test is made in bit positions where C(M) are 0. The tag consists of control bits which may be specified to be either 1 or 0 or may be unspecified. A match test is made on all specified control bits. Contents of the memory matrix and of mask and data registers are unaltered. Match tests are sequenced by the search drive control discussed in Section 1.1.4.

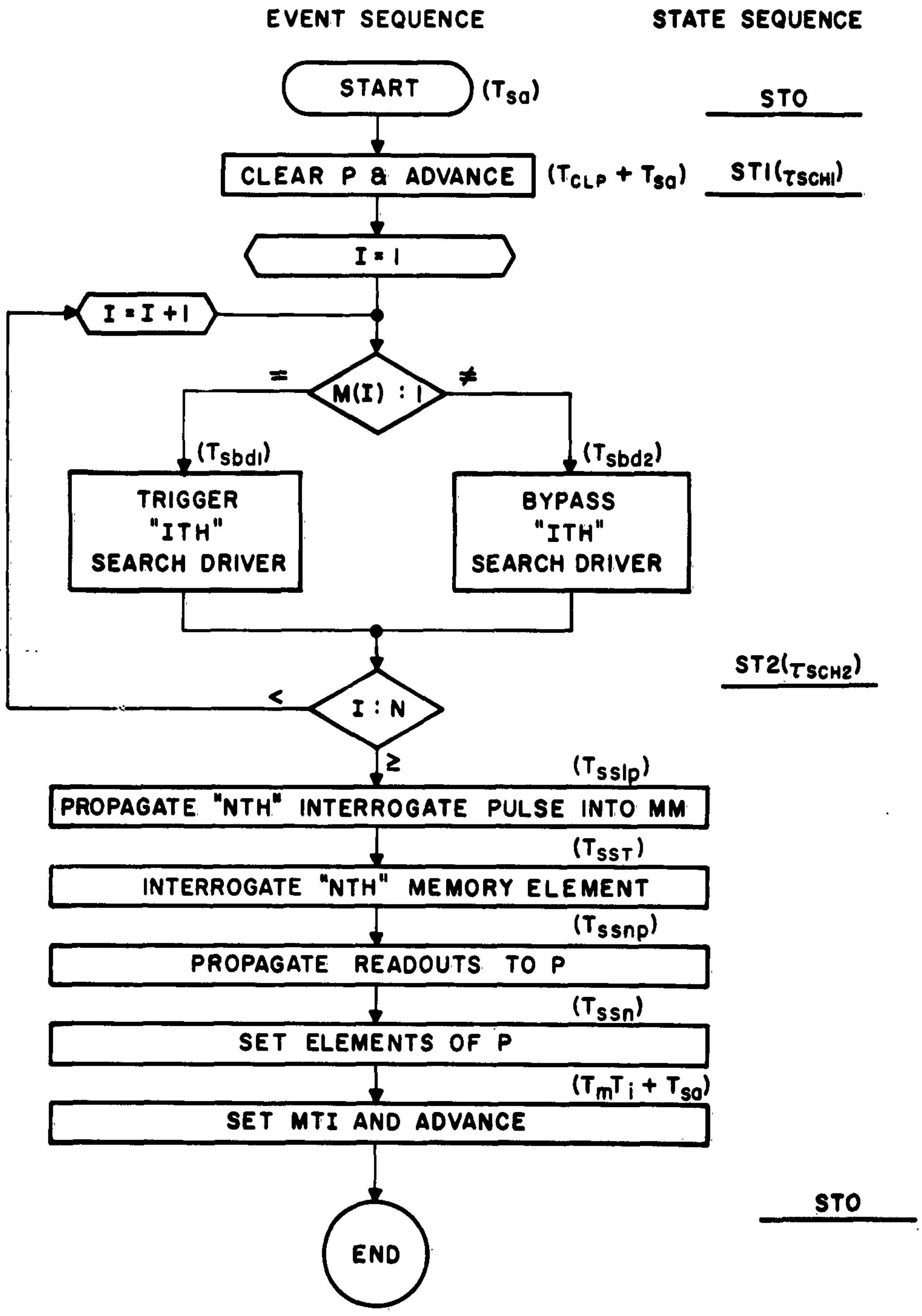
The match detector (P) is initially set to 1 and then is set to 0 in word locations where one or more match tests are not satisfied.

The match type indicator (MTI) indicates the multiplicity of matching words.

The event and state sequences for a SCH command are shown in Flow Chart I. The total time to search a set of words, each containing "n" bits, where the search is performed on k bits ($k \leq n$) is:

$$T_{sch} = \tau_o + \tau_{sch1} + \tau_{sch2}$$

where


EVENT SEQUENCE
STATE SEQUENCE
START
(T_{sa})
ST0
CLEAR P & ADVANCE
($T_{CLP} + T_{sa}$)
ST1(τ_{SCH1})
I = 1
I = I + 1
M(I) : 1
=
≠
(T_{sbd1})
(T_{sbd2})
TRIGGER "ITH" SEARCH DRIVER
BYPASS "ITH" SEARCH DRIVER
ST2(τ_{SCH2})
I : N
<
≥
(T_{sslp})
PROPAGATE "NTH" INTERROGATE PULSE INTO MM
(T_{SST})
INTERROGATE "NTH" MEMORY ELEMENT
(T_{ssnp})
PROPAGATE READOUTS TO P
(T_{ssn})
SET ELEMENTS OF P
($T_m T_i + T_{sa}$)
SET MTI AND ADVANCE
ST0
END

FLOW CHART - I

$$\tau_{sch1} = t_{clp} + t_{sa}$$

$$\tau_{sch2} = k \cdot t_{sbd1} + (n-k)\, t_{sbd2} + t_{sslp} + t_{sat} + t_{ssmp} + t_{ssn} + t_{mti} + t_{sa}$$

t_{sbd1} = interrogation delay per bit

t_{sbd2} = bypass delay per bit

Absolute Maximum Command -- MAXA

The MAXA command locates members of some selected input set which store the maximum absolute numeric value for some specified field and determines the maximum value in this field.

The command is executed over cells having a true detector element. The field over which maximization occurs is defined by the contents of the upper and lower limit registers for some index resistor as described in Section 1.1.4. Bit positions not in the maximization field are not examined. The specified field in each CAM cell is interpreted as an unsigned positionally weighted number having the weight "a_k" associated with the k^{th} bit position.

The index counter stores the index k which runs over bit positions of the specified field starting at the most significant bit. Allowable number systems are subject to the restriction that the weight a_k has a unique value larger than the value of any number generated using only lesser weights. Some number representations which satisfy requirements are:

1) Fixed point binary representation. If numbers are 2's complement, magnitude must be minimized for the set of negative numbers.
2) Fixed point BCD representation.
3) Normalized floating point with leading characteristic.

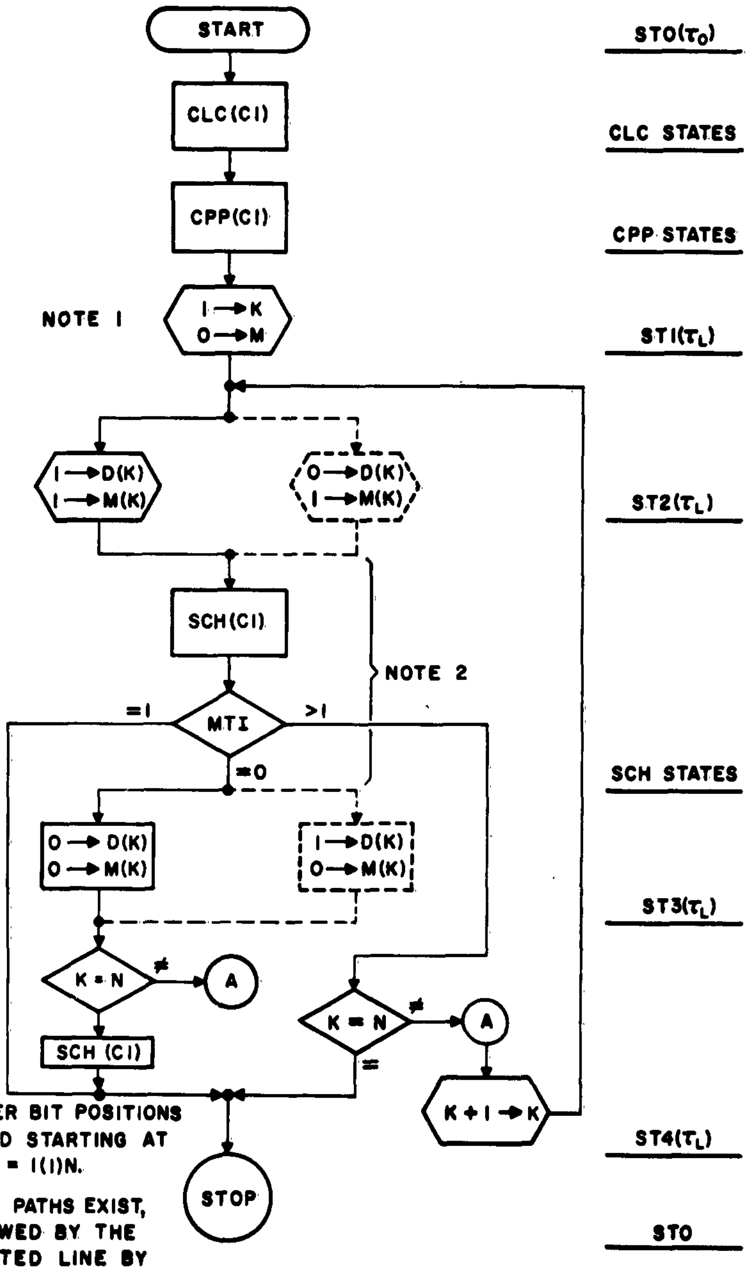
Following execution of a MAXA command, detector elements, P, are "1" for CAM cells in the selected input set which contain the maximum value for the specified field and are "0" elsewhere. The MTI indicates the multiplicity of words having this maximum value for the specified field.

The MAXA command is a compound command, executed as a series of masked equality searches. The unmasked search key (maximum value of the specified field) is generated serial-by-bit starting at the most significant bit. Some "kth" search is executed over the set of selected CAM words which match the key in bit positions 1 through k-1 of the specified field. This set is searched for words having a "1" in the "kth" bit position. If the set satisfying "kth" search has several members, the value of the specified field is taken as "1" in the "kth" bit position and the set satisfying the "kth" search is taken as the input set for the "(k+1)st" search. If the set satisfying the "kth" search is vacant, the maximum value for the specified field is taken as "0" in the "kth" bit positions; this bit position is deleted from subsequent searches, and the input set for the "kth" search is taken as the input set for the "(k+1)st" search. If the set satisfying the "kth" search contains a single member, the algorithm is terminated.

The upper and lower limit registers for the chosen index register are loaded as the MAXA command is issued. The index counter is initially loaded with contents of the upper limit register and is decremented by one for each search of the maximization field. The algorithm is terminated when a unique maximum is located or when contents of the index counter equal the contents of the lower limit register.

The event and state sequence for the MAXA command are illustrated in Flow Chart II. Where a basic command is indicated as an event, the event and state sequence for that command are implied.

Since the MAXA command is terminated at the first step for which a unique maximum is determined, execution time for the command is in general dependent on the distribution of numerical values for words in the selected set as well as on the length of the specified field. The maximum execution time for a MAXA command, executed over a field of length P, occurs when the maximization field is zero for all cells in the selected input set.



NOTE 1. INDEX K RUNS OVER BIT POSITIONS IN THE MAXIMIZATION FIELD STARTING AT THE MOST SIGNIFICANT K = 1(1)N.

NOTE 2. WHERE PARALLEL PATHS EXIST, THE SOLID LINE IS FOLLOWED BY THE MAXA COMMAND, THE DOTTED LINE BY THE MINA COMMAND.

FLOW CHART - II

$$T_{maxa_{max}} = \tau_o + \tau_{clc} + \sum_{i=1}^{2} \tau_{cppi} + (P+1) \sum_{i=1}^{3} \tau_{schi} + (3P+1)\, \tau_1$$

(4P + 8 states)

The minimum execution time for a MAXA command, executed over a field of length P, occurs when a unique maximum is located at the first search and is:

$$T_{maxa_{min}} = \tau_o + \tau_{clc} + \sum_{i=1}^{2} \tau_{cppi} + \sum_{i=1}^{3} \tau_{schi} + 2\, \tau_1$$

(9 states)

REFERENCES TO CHAPTER I - SECTION I-1

1. Slade, A. E., and H. O. McMahon, "A Cryotron Catalog Memory System", Proc. of EJCC, pp. 115-120, December, 1956.

2. Slade, A. E., and C. R. Smallman, "Thin Film Cryotron Catalog Memory", Symposium on Superconductive Techniques for Computing Systems, Washington, D. C., May, 1960.

3. Frei, E. H., and J. Goldberg, "A Method for Resolving Responses in a Parallel Search File", IRE Trans. on Electronic Computers, Vol EC 10, No. 4, pp. 718-722, December, 1961.

4. Kiseda, J. R., H. E. Petersen, W. C. Seelbach and M. Teis, "A Magnetic Associative Memory", IBM Journal of Research and Development, Vol. 5, No. 2, pp. 106-121, April 1961.

5. Seeber, R. R., "Cryogenic Associative Memory", National Conference of the ACM, Milwaukee, August, 1960.

6. Seeber, R. R., "Associative Self-Sorting Memory", Proc. of EJCC, pp. 13-15, December, 1960.

7. Seeber, R. R., and A. B. Lindquist, "Associative Memory with Ordered Retrieval", IBM Journal, January, 1962.

8. McDermid, W. L., and H. E. Petersen, "A Magnetic Associative Memory System", IBM Journal of Research and Development, Volume 5, No. 1, pp. 59-62, Jan. 1961.

9. Minnick, Robert C., "Magnetic Comparitors and Code Converters", Symposium on the Application of Switching Theory in Space Technology, Sunnyvale, Calif., Feb. 1962.

10. Davies, P., "A Superconductive Associative Memory", SJCC, San Francisco, May, 1962.

11. Newhouse, Vil and R. E. Fruin, "A Cryogenic Data Addressed Memory", SJCC, San Francisco, May, 1962.

12. Rosin, Robert F., "An Organization of an Associate Cryogenic Computer", SJCC, San Francisco, May, 1962.

13. Kolmogovov, A. W., and S. V. Fomin, Elements of the Theory of Functions and Functional Analysis, Graylock Press, 1957.

14. Estrin, G., "Organization of Computer Systems--The Fixed Plus Variable Structure Computer", EJCC, San Francisco, May, 1960.

I-2 Assignment Problems

The concept of a fixed plus variable structure computer system, (F+V) was proposed in 1959 by G. Estrin,[1, 2] at UCLA. The primary goal of the F+V system organization, as stated by Estrin, is: "to permit computations which are beyond the capabilities of present systems by providing an inventory of high speed substructures and rules for interconnecting them, such that the entire system may be temporarily distorted into a problem-oriented special purpose computer". Equivalently, one could state that the goal of the F+V system organization is to extend the class of practicably computable problems, where "practicable computability" of a problem, is primarily a function of programming time, computation time, and cost of computing.

The F+V system aims to achieve the above goals by reducing the computation time of a problem as a result of a combination of the computational power and flexibility of a general purpose computer (F) with the speed of simultaneously operating special purpose structures (V). The general purpose (GP) computer is referred to as "fixed" in the sense that its hardware has been connected into a structure which is expected to vary very little during the life-time of the computer, and thus does not require specification of the interconnections of its elements along with the statement of the problem. This property permits development of programming languages for the GP computer resulting in easier communication between the user and the machine at some expense in pure computational speed.

The role of F in the F+V system is, in addition to being a powerful computing unit, to contain the complex, though not in general most time consuming parts of the program for the problem, perform most of the input-output operations, and to permit the use of already developed higher programming languages as well as the use of large program and subroutine libraries associated with a reliable modern GP computer.

The variable structure computer, V, is used to gain speed in computation by employing special purpose computation techniques or generally less complex iterative procedures. The variable nature of the interconnections of its hardware permits utilization of the same hardware for different structures, a necessary condition to make the F+V system economically feasible. The more common of the special purpose techniques to be used in V structures are wired programs, and problem-oriented organizations of the hardware which may utilize, wherever effective, unconventional interconnection of computing circuits and unconventional number representation.

In order to combine the high speed of special purpose computers with flexibility required in efficient computation, real time structure changes in V must be permitted.

These structure changes may be effected by electronic or electromechanical switching of the hardware into preconstructed structures, or by actual physical change in the interconnections of V hardware. The latter changes are usually quite time-consuming, even though most of these may only involve substitution of units previously constructed off-machine. Nevertheless, such a mechanical change in V structure may result in considerable reduction of overall computation time and may be the deciding factor in rendering a problem practicably computable.

Equivalent to a sub-routine library in a GP computer, V will have a library of substructures of more common functions. Some of the more frequently used structures will usually be electronically switchable in the structure for a given problem.

Whenever it is not distorted into a form for application to a particular problem, V is in a "standard state" configuration. In the standard state a number of electronically switchable configurations are available for efficient performance of more common elementary operations or for computation of elementary functions. In this state V may be considered as an extension of F for computing these functions as macro-commands, or it may be used to execute independent programs.

In a non-trivial application of the F+V system, F and one or more configurations of V (constrained by the size of its inventory) will compute in parallel.

Their interaction is controlled by a separate supervisory control unit, SC, which enforces cooperation or synchronization of processes going on in F and V. The SC monitors operations in F and V and arranges for interlocks and data transfers between these. It is in direct communication with V via an appropriate set of control signals, and uses interrupt features and a set of special commands of F to communicate with the latter. At this stage, the control functions of the SC will be kept as simple as possible; in concept, however, it could conceptually involve a complex computer in its own right.

The characteristics of F, V and SC are discussed in detail below. A block diagram of the F+V system is given in Figure 1. 5.

1. 2. 1 The Fixed Structure Computer, F.

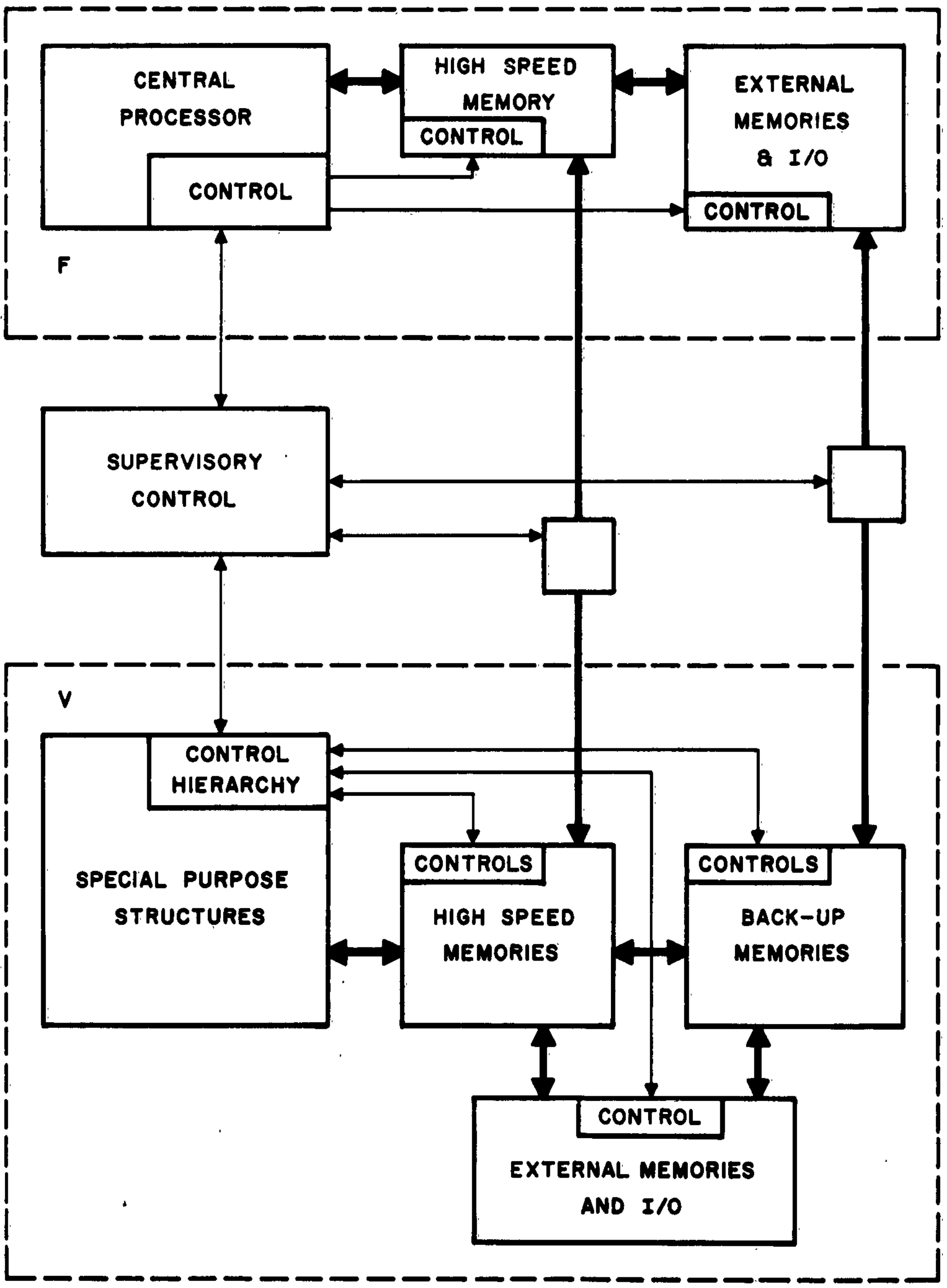
Some of the reasons for including a fixed structure GP computer in the F+V system were discussed in the previous section. Briefly, all parts of the problem where no significant gain in speed can be expected through special purpose techniques will be left for computation in F. V will be used to mechanize only these parts of the problem which require the major (generally iterative) portions of its overall computing time.

Similarly, all operations which F is better equipped to handle, such as most of the input-output operations, will be delegated to it.

Examples of analysis of complex problems and assignment of tasks to F and V are to be found in References 3, 4, 5, 6, 7, 8 and 9.

The first choice for F was the IBM 7090 solid state computer. This choice was based on the fact that the IBM 7090 is a modern fast computer with a large and versatile list of instructions, independent input-output processing, a large number of auxiliary storage and input-output units, and readily available (i.e., already on UCLA campus). The FORTRAN programming language is convenient to use, and a large number of programs and subroutines are available from many 7090 users through the SHARE organization. Further detailed information about the characteristics of IBM 7090 can be obtained from Ref. 10.

In the following discussion any reference to F implicitly refers to the IBM 7090.



BLOCK DIAGRAM OF THE F+V SYSTEM

FIGURE 1. 5

1.2.2 The Variable Structure Computer, V.

It was already stated that V is a collection of hardware, usually in the so-called "standard state", which can be restructured into problem-oriented special purpose structures. The principal special purpose techniques to be used are wired program control, specialized computing circuits, and unusual number representation, all of which usually cannot be incorporated directly in a GP computer structure, and which are feasible in V only due to sharing of hardware between different structures.

During a computation V may be in the standard state or in one or several problem-oriented structures, and it may change structure several times before the computation is completed. The structure changes may be effected by electronic or electro-mechanical switching, or by actual physical rearrangement of interconnections. Each structure contains its own control units which communicate with a higher order control unit.

1.2.3 The Standard State

In the standard state the current inventory of V is used to increase the instruction list of F with more commonly used operations. These operations can be performed in V using any unconventional methods which permit faster execution of the operation.

Among the commonly used operations which may be included in the standard state of V are such elementary functions as trigonometric functions, inverse trigonometric functions, logarithmic and exponential functions, n-th root or power, hyperbolic functions, Bessel functions, differentiation and integration, random number generation, and statistical operations; complex arithmetic, vector arithmetic; matrix operations; and non-arithmetic operations. Some of the listed operations may be available in single, multiple, or partial precision. Simultaneous computation of certain subsets of the above operations may be incorporated in the standard state. For a given operation a choice between high-speed or low-speed structures which require large or small amounts of hardware, respectively, may be available.

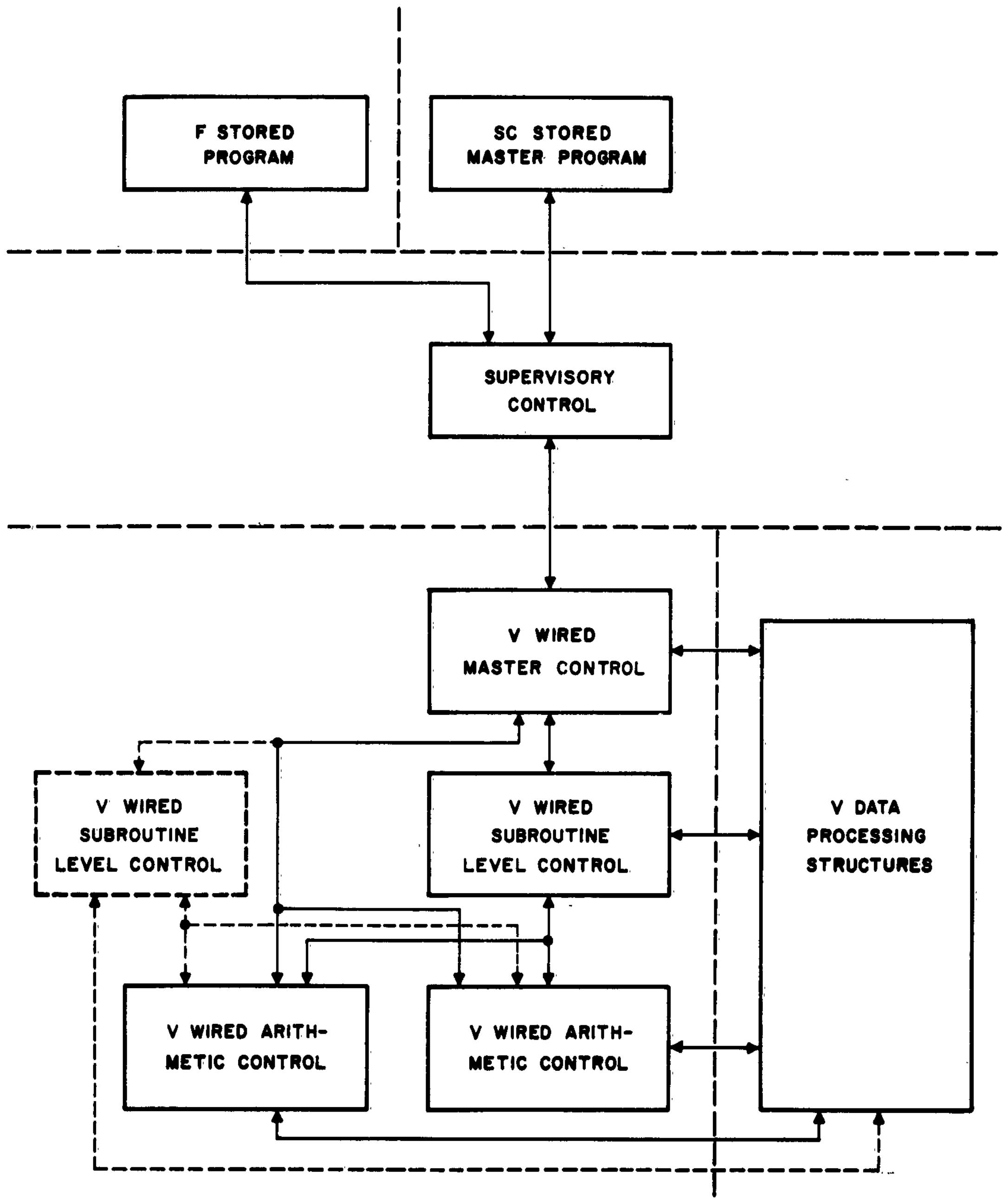
If included in the standard state, such operations can be performed by electronic switching of the standard state control unit(s), and thus very rapidly available. The number of such operations actually included in the standard state is a function of available inventory and may be small in the initial standard state. However, as V inventory is increased, the power of the standard state is increased accordingly.

Based on an extrapolation of the hardware requirements established during problem studies (see reference above) the V hardware inventory is eventually expected to permit construction of the following simultaneously operating units: a 144 bit parallel arithmetic unit, 360 bits of shifting registers, 144 bits of comparators, 27 bits of counters, and control units containing 1000 logical elements. In addition, approximately 8 simultaneously accessible very high speed memory units (0.5 μsec. access time) of 1000 72-bit words each, 2 simultaneously accessible 16,000 72-bit high speed (1 μsec. access time) memory units, a 1000 word content addressable memory, and a large slower back-up memory (e.g., a disc memory unit) are expected to be contained in V.

1.2.4 The Control Hierarchy

The execution of computations in V is controlled by a number of control units. Since some of the control units can exercise control over others, a control hierarchy (or control tree) is formed. Each of the control units in the hierarchy, in general, contains circuitry for electronic switching of its wired program such that it may be used to control several different programs. Use of stored program control units, however, is not entirely ruled out.

A typical configuration of the control tree is depicted in Figure 1.6. On the lowest control level are control units which control such common arithmetic and logical operations as addition, subtraction, multiplication, division, square root, floating point to fixed point and vice versa conversion, and possibly others. Each of the arithmetic operations may be performed in fixed or floating point, single, double or partial precision. The exact nature of such a control unit depends on the arithmetic schemes used but, in general,


F STORED PROGRAM
SC STORED MASTER PROGRAM
SUPERVISORY CONTROL
V WIRED MASTER CONTROL
V WIRED SUBROUTINE LEVEL CONTROL
V WIRED SUBROUTINE LEVEL CONTROL
V DATA PROCESSING STRUCTURES
V WIRED ARITH-METIC CONTROL
V WIRED ARITH-METIC CONTROL

F+V CONTROL HIERARCHY

FIGURE 1.6

it corresponds in complexity to the arithmetic control unit of a large scale general purpose computer. Control units on the lowest control level can generate only elementary commands, i.e., commands which cause only a single action to take place (e.g., a transfer from one register to another, a shift, counting by 1). The particular operation executed by such a control unit is specified by a compound command generated by a control unit on a higher control level.

On the next control level are so-called "subroutine" control units. These control units may execute elementary functions, complex arithmetic, vector algebra, etc. Each control unit on this control level may use the operations controlled by lower level control units as compound commands (e.g., may generate control signals which specify "add, single precision, floating point", etc.), and also generate all elementary commands. The wired programs of a subroutine control unit usually correspond to a stored program subroutine in a general purpose computer.

A number of consecutively higher control levels may exist, each of which may specify operations performed by lower level control units as compound commands. Some of these control units may use stored programs. On the highest control level, however, is the supervisory control unit.

The purpose of the supervisory control is to supervise the execution of the computations in F and in V, to coordinate information exchanges between F and V; between F or V and peripheral equipment, and between the latter themselves; and to perform interlocking functions. For efficient performance of this task, the supervisory control unit may contain a wired program along with stored programs in both F and V.

Each of the control units, regardless of the control level, has to perform three basic functions: (1) generate a sequence of states which correspond to sequential operations specified by the program being mechanized, (2) during each sequence state, generate a subset of available elementary and compound commands, where the commands in the subset correspond to those operations in the program which may be performed simultaneously, (3) in each sequence state provide for generation of the next sequence state as specified by the program. The next sequence state may be different from the natural sequence due to unconditional or conditional branching specified by the program.

Of the enumerated functions, only the first can be problem-independent if a large enough number of sequence states is available. The other two, necessarily, depend on the algorithms being mechanized and thus represent what is equivalent to a stored program in a general purpose computer.

Consequently, for a fixed hardware structure (as is the case for computing many types of elementary functions where the hardware is in the usual arithmetic unit configuration) a change in the computational algorithm (program) implies a change in the command generating structure as, in general, different commands are generated in the same sequence state for different algorithms, and a change in the branching structure as different algorithms require different branching. These changes in the control unit will, most likely, be effected by mechanical means if the control structure is on a sufficiently high level of the control hierarchy, and by electronic means on the subroutine and arithmetic operation levels.

There are several courses of action open for changing of the branching logic and command matrix structures. A very general approach would involve establishing branching and command matrices which permit transition from a given sequence state to any other sequence state for any one of the completion signals and control register states, and generation of any command for a given sequence state and state of the control register, respectively. Given such matrices, a wired program could be established by enabling proper subsets of intersections of the branching and command matrices under the control of a plug-board, punched card, or a photo-electric switching device.

The major advantage of such a general switching scheme is the speed of restructuring of the control unit for different uses of the F+V system and the low cost of doing so. A disadvantage is that the initial cost of such a control unit, in terms of hardware, may be very large. Studies of control requirements of a number of elementary functions have also indicated that only a small percentage of the total intersections of such a matrix are enabled for a given problem. Hence a large portion of the V hardware may stand idle. Further, the speed of switching from one program to another during a given problem, even if it is high in terms of mechanical change, may be considerably slower than could be done electronically. Finally, the required large matrices of switching circuits may slow down the operations of the control unit during executing a particular wired program.

Another course of action involves building M branching logic units and M command matrices to correspond with the M electronically switchable wired programs which are to be associated with the given control unit. Each of the matrices would contain only the intersections required by the program mechanized and switching from one wired program to another can be accomplished at electronic speeds. A mechanical structure change of the control, however, will be time consuming as new branching logic units and command matrices need to be wired.

On the basis of the large hardware requirements and low inventory utilization, the proposed control unit will use the second approach, i.e., a separate branching logic unit and command matrix for each of the electronically switchable wired programs. Use of general matrices in some of the control units in the future is, however, by no means ruled out.

Since the control functions of control units on all levels of control are the same, each such control unit may be mechanized by using the same basic model of the control unit. The structure and logical properties of such a model are discussed in the following section.

1.2.5 A Model of the Control Unit

In this section a model of a control unit with electronically switchable wired programs will be described. This control unit may be used in all levels of the control hierarchy. Let V(i) be the i^{th} special purpose structure of V, then CU(i, j) designates the j^{th} control unit of V(i). When explicit reference to V(i) has been made, however, it is sufficient to designate CU(i, j) by CU(j).

This simplified notation is used in the sequel, i.e., the index i is not used.

The logical design of the proposed control unit is based on asynchronous design philisophy. There are several versions of the latter. Totally asynchronous design, as formalized by Muller and Bartky,[11] requires that the correct completion of a control operation is verified before the next control operation is permitted to start. For example, completion signals are generated at every memory element of a register to indicate that the outputs of a particular memory element correspond to its inputs.

A less rigorous asynchronous design method requires positive verification of completion of the control operation only at a few points of the circuitry. Finally, completion signals may be generated without actual verification of the correct completion of the control operation. Such completion signals are based on elapsing of a time interval which is known to be sufficient for correct completion of the control operation. A relatively complete bibiliograpy on asynchronous design is given in Reference 12.

Totally asynchronous circuits permit very fast operation (e.g., as soon as all the circuits involved have responded to a control signal and the proper completion signals have been formed, the state of the control may be changed to generate a new control signal) and their operation is reliable, but hardware requirements of the completion signal circuits are rather large and thus a major portion of the V hardware would not be available for actual computing operations.

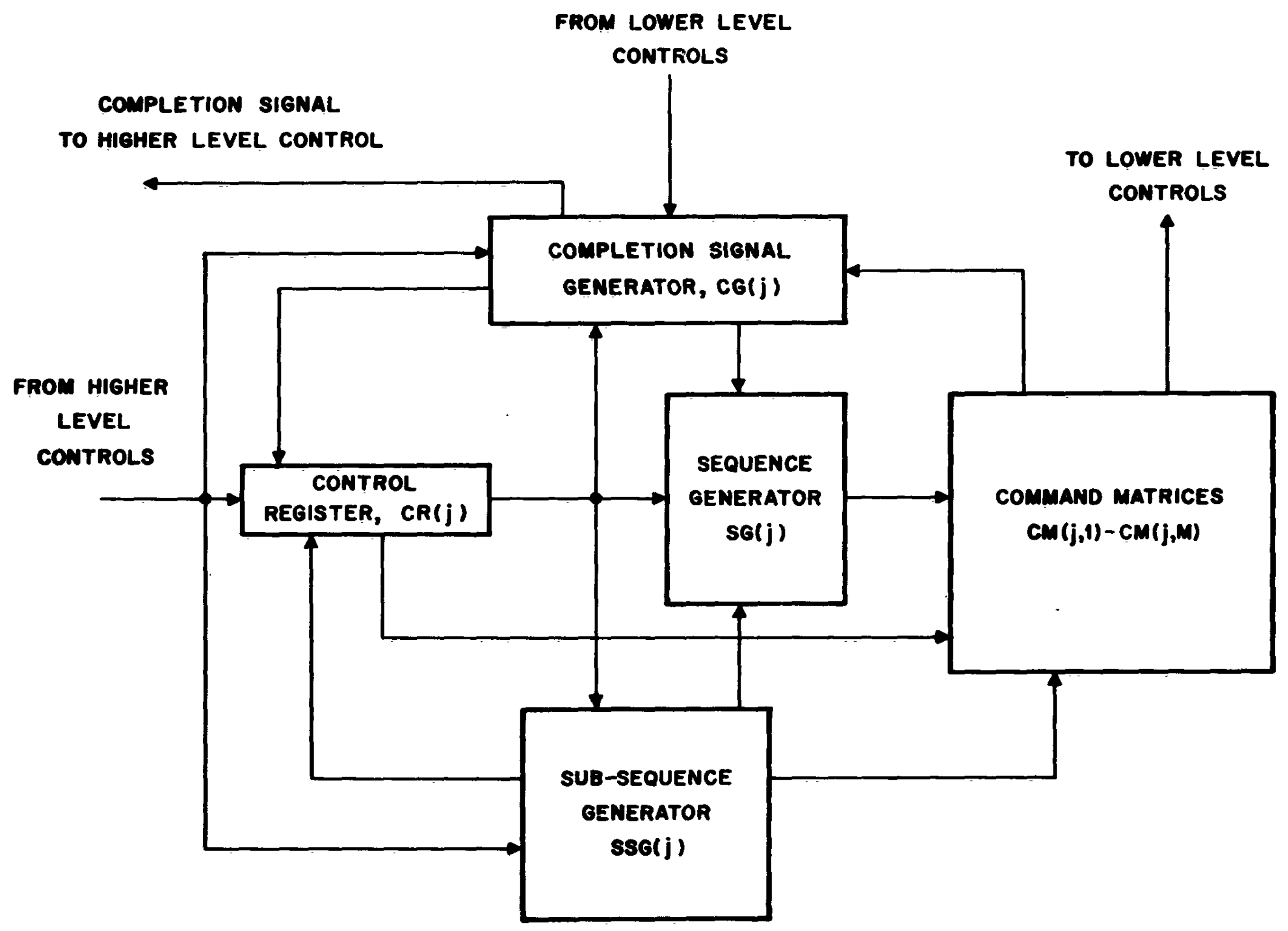
The next level of asynchronism (not all of the circuit points are tested for actual response to the control signal) permits fast operation, but the correct response of all the circuitry to the control signal is no longer guaranteed. Hardware requirements have been reduced but may still be considerable. The last discussed level of asynchronism (simulation of the operation times of circuits without actual test for any response to the control signal) offers no indication of correct operation of the circuits and the time elapsed before the completion signal is generated must allow for the worst case variations of the response times of the circuits involved in the control operation. Hardware requirements, however, are small.

The proposed control unit will utilize the last two types of asynchronous design. The totally asynchronous was rejected on the basis of large hardware requirements. The elementary commands will be partitioned into classes such that the operation times of the elementary commands in a given class are almost equal. For each such class, a single completion signal is generated after a time interval sufficient for completion of commands in the class has elapsed. Completion signals for compound commands are generated by the corresponding lower level control units as soon as the execution of the command has been completed. Elementary commands which perform comparison or test operations will have at least two completion signals, that is, the completion signals are used to indicate the result of the compare or test operation.

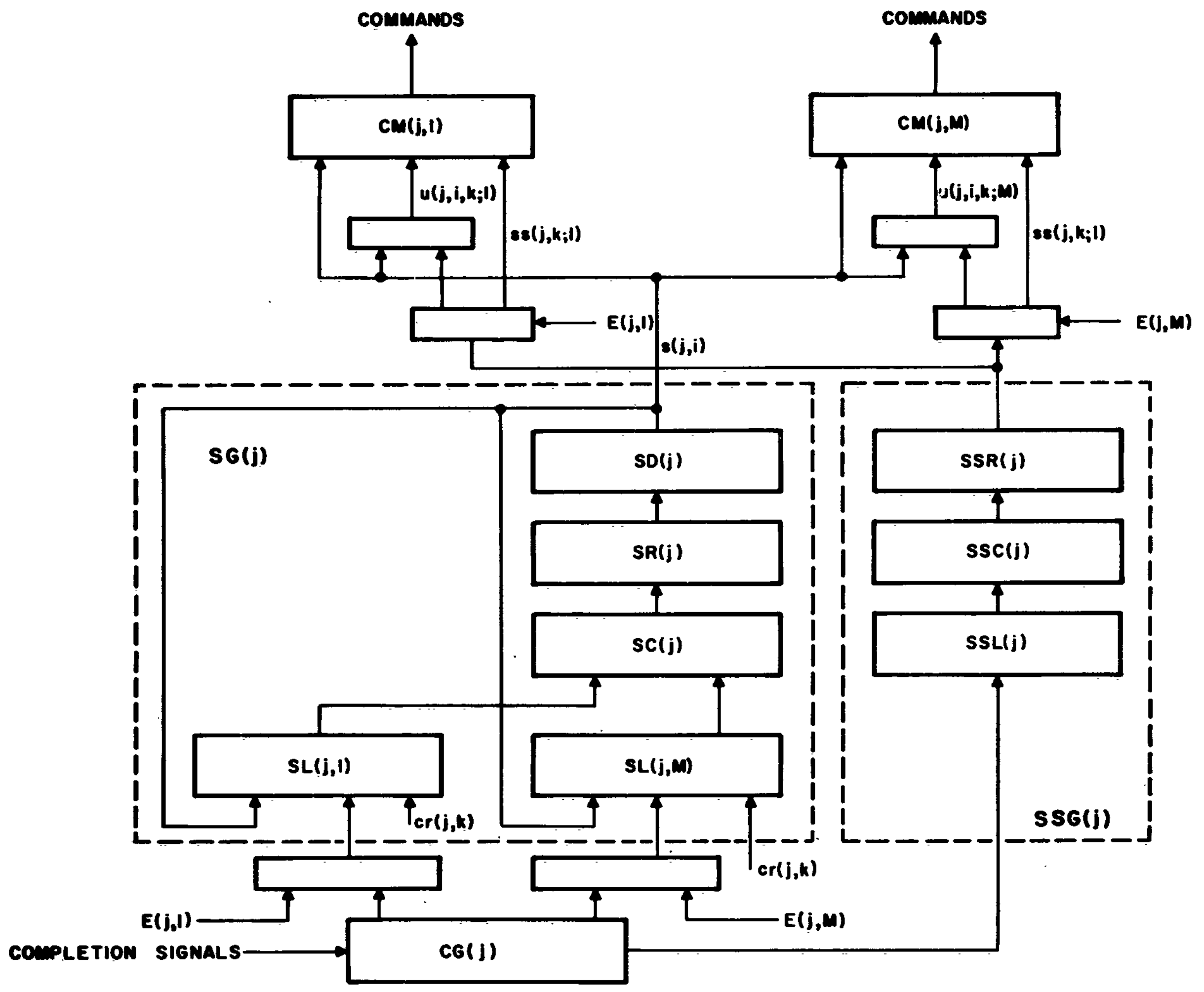
Figure 1.7 depicts the block diagram of the proposed control unit, CU(j). The major units of CU(j) are: a control register, CR(j); a sequence generator, SG(j); a subsequence generator, SSG(j); command matrices, CM(j, m); a completion signal generator, CG(j); and a timing chain, TC(i). Figure 1.8 depicts these units in more detail. The purpose and structure of each unit, as well as their interaction, will be discussed in detail below.

The control register, CR(j), is a collection of k memory elements, CR(j, k), which are used to store information concerning the particular computation in progress, results of tests and comparisons which may be utilized later in the computations, etc. The outputs of CR(j, k) are designated as cr(j, k;1) and cr(j, k;0), to represent the "1" and the "0" states of CR(j, k), respectively. The outputs of CR(j) may be used in the branching logic of the sequence generator or in the control matrices.

The sequence generator, SG(j), contains a sequence counter, SC(j); a sequence register, SR(j); and M branching logic units, BL(j, m), corresponding to the M different wired programs associated with CU(j). The purpose of the sequence generator is similar to the program counter in a general purpose computer, i.e., to permit execution of the different computational steps in a predetermined sequence. The sequence counter, SC(j), generates as many as 2^n sequence steps, corresponding to its n memory elements. The current state


FROM LOWER LEVEL CONTROLS
COMPLETION SIGNAL TO HIGHER LEVEL CONTROL
TO LOWER LEVEL CONTROLS
COMPLETION SIGNAL GENERATOR, CG(j)
FROM HIGHER LEVEL CONTROLS
CONTROL REGISTER, CR(j)
SEQUENCE GENERATOR SG(j)
COMMAND MATRICES CM(j,1)-CM(j,M)
SUB-SEQUENCE GENERATOR SSG(j)

BLOCK DIAGRAM OF CU(j)
FIGURE 1.7



CONTROL UNIT CU(j)

FIGURE 1. 8

of SC(j) is stored in the sequence register, SR(j), and the sequence counter advanced to its next state as prescribed by the branching logic of the particular program being executed. In the case where the next state depends on the result of a compare or test operation, the next state is based on the assumption that the test or comparison failed (was "false"). If the test is actually "true", SC(j) is set into the required state as soon as the result is known. This type of operation is successful if the results of comparing and test operations which have the higher probability of occurring are labelled as "false".

The memory elements of the sequence counter, SC(j), are denoted as SC(j, 1) through SC(j, n), corresponding to n memory elements; their outputs are denoted by SC(j, k;1) and SC(j, k;0), for the kth memory element. The elements of the sequence register, SR(j), are denoted by SR(j, 1) through SR(j, n). The outputs of the kth memory element are denoted by sr(j, k;1) and sc(j, k;0).

The outputs of SR(j) are combined in a decoder, SD(j), into 2^n sequence states, s(j, 1) through s(j, 2^n). Only those sequence states which are required for a particular wired program are generated.

The M branching logic units, SL(j,m), use information from the sequence decoder, control register, and completion signals to advance the sequence counter into its next state. The outputs of the ith branching logic unit are the ith set, reset, and trigger signals for the R-S-T type memory elements of the sequence counter, i.e., s(i)-SC(j, k), r(i)-SC(j, k), and t(i)-SC(j, k), for the kth memory element of SC(j).

The subsequence generator, SSG(j), contains a subsequence counter, SSC(j); a subsequence register, SSR(j); and logical circuits, SSL(j), for advancing SSC(j) into its next state. For each state of the sequence generator, the subsequence generator may be advanced through p successive states, corresponding to the p memory elements in the ring counter type structure of SSC(j). The reason for inclusion of SSG(j) is to reduce the number of states required in SSG(j) and, correspondingly, to reduce the complexity of the M branching logic units. The states of SSC(j) occur always in a fixed order (no branching) and thus are used to sequence parts of the wired program which involve no branching. The SSC(j) may be reset at any time to its origin state. This is done every time the state of the sequence generator is changed.

The memory elements of SSC(j) and SSR(j), as well as their outputs, are denoted SSC(j, k) and SSR(j, k), and ssc(j, k, 1), ssc(j, k, 0), ssr(j, k, 1), and ssr(j, k, o), respectively.

The selection of the mth command matrix, CM(j, m), for a wired program, E(j, m), is accomplished by generating M complete sets of subsequence states, ss(j, 1;m) through ss(j, k;M), corresponding to the wired programs. The outputs of this structure switching unit go to all of the M command matrices.

The outputs s(j, i), of the sequence generator, and the outputs ss(j, k;m), of the subsequence generator, may further be combined (for circuit economy reasons) by logical AND function into signals denoted as u(j, i, k;m).

The M command matrices, CM(j, m), generate the elementary and compound commands which are specified for a given subsequence state of a given sequence state. As was mentioned before, the set of elementary commands may be partitioned into classes consisting of elementary commands whose operation times are approximately equal. A particular partition proposed here is: parallel data transfer commands, mx(j, k); test and compare commands, mt(j, k); memory read command for each memory, me(j, k); miscellaneous commands, mz(j, k) which include setting and resetting of registers or memory elements, and counting by one; compound elementary commands, (arithmetic commands) ma(j, k); and modifiers of the compound commands, h(j, k), which specify whether the compound command to be executed is single precision, double precision, fixed point, floating point, etc. As the number of different elementary commands increases, other partition schemes may be required.

The commands generated for a wired program are specified by combining, using logical AND functions, the required sequence state, s(j, k), and the subsequence state of the given wired program, E(j, m), ss(j, 1;m).

The completion signals for the elementary commands are generated in the completion signal generator, CG(j), where proper time delay units are used. Completion signals for compound commands are generated in the proper lower level control units. Completion signals are designated by cx(j) for the class of transfer commands, cz(j), and ce(j, k) for the miscellaneous elementary commands and for the kth memory read command; ca(j, k) for the kth compound command; and ct(j, k;r) for the rth result of ct(j, k). All completion signals are combined with the wired program selection signals, E(j, m), to generate a complete set of completion signals for each wired program such that selection of the proper branching logic unit is done by the appropriate set of completion signals.

Finally, the control of the control unit itself is executed by a timing chain, TC(j), which generates signals for transferring the contents of the sequence and subsequence counters into respective registers, advancing the states of the former, and correcting the states in the case where the assumption of "false" test result was assumed.

A detailed procedure for designing control units for mechanizing computation of elementary functions, as well as a number of examples of such control units are given in Reference 13.

A convenient description of the operation of the control unit for mechanizing a given computation can be obtained by constructing a table which shows the "present" and the "next" states of the control unit. A particular "present state" entry specifies the states of the sequence generator, S(j, k), subsequence generator, SS(j, k;m), and the control register, CR(j, i); lists the commands generated, and shows the states of the sequence generator, and the control register as functions of the completion signal, i.e., the "next state". An example of such a description is given in Figure 1.9.

1.2.6 Description of V Structures

In order to develop systematic procedures for designing V structures, it is necessary to establish a framework for systematic description of such

Present state					Next state	
S (j, k)	ss (j, i;m)	CR(j, i) 123456789	Commands	Completion signals	s (j, k)	CR(j, i) 123456789
...	...	...	...	...	...	...
1	1		mx(j, 2)			
			mx(j, 3)			
		--------0	mz(j, 1)			
		--------1	mx(j, 4)			
		--------1	mz(j, 2)			
	2		mx(j, 5)			
			mx(j, 6)			
			mx(j, 7)			
	3	--------1	mx(j, 8)			
		--------1	mz(j, 3)			
	4		mx(j, 9)			
			mx(j, 11)			
		--------1	mx(j, 5)			
			mt(j, 1)			
		--0------		cmt(j, 1;1)	10	
		--1------		cmt(j, 1;1)	2	
		-01------		cmt(j, 1;1)	2	----0----
				cmt(j, 1;2)	2	
		-01------		cmt(j, 1;2)	2	----0----
...	...	...	...	...	...	...

AN EXAMPLE OF DESCRIPTION OF OPERATION
OF A CONTROL STRUCTURE

FIGURE 1. 9

structures. As a part of this framework, the concepts of macro-description and micro-description of V structures are formulated.

1.2.7 Macro-description

A macro-description of a V structure will specify the following important large-scale features of the structure:

1. Computational characteristics in terms of the compound commands available to the supervisory control. For each such command, the upper and lower bounds and the averages of their computation times, % hardware used exclusively, and % hardware time shared are given.
2. The inventory involved in terms of functional units, such as control units, arithmetic units, memory units (including any input-output units), registers, combinatorial logic units, non-digital devices, etc.
3. The logical interconnections between the functional units.
4. The change from standard state in terms of functional units.
5. The cost associated with the change from standard state, in terms of time, hardware, and other significant parameters (if any).
6. Computational history of the structure in terms of problems in which used, how many times, how long, errors, etc.

1.2.8 Micro-description

A micro-description specifies the detailed design of the V structure as follows:

1. The elementary commands and the elementary compound commands are listed.
2. Computational algorithms for each of the compound commands available to the supervisory control are expressed in mathematical terms.
3. Designs of control unit structures for executing compound commands are given.

4. Computation times of the compound commands are given in terms of lower order execution times, until all times are in terms of a set of elementary operation times.

5. Functional units are described in terms of basic amplifier modules and logic (combinatorial) modules. The extent of fan-in and fan-out of each basic module is given. Other types of modules may be defined as need arises.

6. Logical interconnections of the basic modules are given.

7. Locations of basic modules are given referred to an adequate co-ordinate system (co-ordinates may refer to frames, motherboards, locations on motherboards, etc.).

8. Physical interconnections of basic modules are given in an adequate co-ordinate system (coordinates may refer to connectors, pins; types of connecting wires, lengths of connecting wires may be specified).

9. A change from the standard state is given in terms of change in the requirement for different types of basic modules, change in the type of the basic module at a given location, and in terms of change of physical interconnections between basic modules.

10. The cost of a change from the standard state is given in terms of costs of change of lower order units of the V structure, until the costs are expressed in terms of a set of elementary cost units.

1.2.9 Notation for Macro-description

Let $V(i, k; p_1; p_2; p_3; p_4; p_5; p_6)$ represent the i^{th} configuration of the V inventory, referred to standard state k. The p_j refer to different properties of $V(i, k)$.

1.2.9.1 Computational characteristics of V(i); property p_1

The computational characteristics in a macro-description of $V(i, k)$ are expressed as a list of the compound commands available to the supervisory control. For each of the compound commands its computing times (maximum, average, and minimum) and V hardware utilization (exclusive and shared) are given.

$$V(i;\, p_1) = \left\{ \left(f(k, c;q_i),\ T\text{-}f(k, c;q_i)\ H\text{-}f(k, c;q_i),\ P\text{-}f(k, c;q_i) \right) \right\}$$

where

$k = 1, \ldots K,\ i = 1, \ldots, I.$

$f(k, c;q_i) = k^{th}$ compound command of class c available to the supervisory control. The property q_i refers to the i^{th} computational algorithm of f(k, c) and to the design of the corresponding control unit.

$T\text{-}f(k, c;q_i) = \left(Tma\text{-}f(k, c;q_i),\ Tav\text{-}f(k, c;q_i),\ Tmi\text{-}f(k, c;q_i)\right)$ specifies the maximum, average, and minimum computation times of $f(k, c;q_i)$ respectively.

$I\text{-}f(k, c;q_i) = \left(Ie\text{-}f(k, c;q_i),\ Is\text{-}f(k, c;q_i)\right)$ specifies the amount of inventory utilized by $f(k, c;q_i)$ exclusively and shared, respectively.

$C\text{-}f(k, c;q_i) = \{c_j\}$, specifies the classes of compound commands which may be executed simultaneously with $f(k, c;q_i)$.

1.2.9.2 Description of inventory: property p_2

The inventory of V(i, k) is described in terms of functional units: control units, arithmetic units, registers, combinatorial logic units, memories, and non-digital devices. In the following notation the indices, i, k are omitted for simplicity.

$$V(i, k;q_2) = \left(\left\{CU(j;p_k)\right\}, \left\{AU(j;p_k)\right\}, \left\{RU(j;p_k)\right\}, \left\{LU(j;p_k)\right\}, \left\{MU(j;p_k)\right\}, \left\{NU(j;p_k)\right\} \right)$$

$CU(j;p_k)$: j^{th} control unit with property p_k. Property p_k specifies the compound commands mechanized by CU(j), the logical design of the control unit, the amount of hardware utilized, and the fan-in and fan-out distributions.

$AU(j;p_k)$: j^{th} arithmetic unit with property p_k. Property p_k specifies the arithmetic operations performed by AU(j), its logical design, hardware requirements, and fan-in, fan-out distributions.

$RU(j;p_k)$: j^{th} register unit with property p_k. Property p_k specifies the size of the register in bits, its shifting or counting capabilities, hardware required, and fan-in and fan-out distributions.

$LU(j:p_k)$: j^{th} combinatoric logic unit with property p_k, where p_k specifies the number of parallel inputs, nature of the logical operations performed, hardware required, and fan-in and fan-out distributions.

$MU(j;p_k)$: j^{th} memory unit with property p_k, where p_k specifies the type of memory (or input-output unit), its size, access and cycle times, priority structure, special properties (such as content addressable memory, disc file).

$NU(i, j; p_k)$: j^{th} non-digital unit with property p_k, where p_k describes the nature of the non-digital unit.

1.2.9.3 Logical interconnections between functional units: property p_3

Logical connections between functional units specify the required data paths and control lines. Interconnections are specified at the "bit" level of the data handling functional units (registers, etc.) along with the control signal if the particular data path is gated.

Let $FU(j;p_k)$ represent a general functional unit. Interconnections between functional units can be specified as:

$$V(i;p_3) = \left\{ FU(j;r-s) \rightarrow FU(k;t-v) : mx(u) \right\}$$

where a particular element of the set states that the bits r through s of FU(j) are connected or transferred to the bits t through v of FU(k) under the control of the elementary transfer command mx(u).

1.2.9.4 Functional unit level change from the standard state: property p_4

The standard state, V(k), will be considered as a reference frame for changes in V structure. The macro-description of a change of V(i, k) from V(k) is given in terms of functional units and parts of functional units of V(k).

$$V(i;p_4) = \left\{ \left(FU(i,j) = FU(k,m;r-s) + FU(k,m+1,u-v) + \ldots \right) \right\}$$

which state that FU(i, j) is composed of bits r through s of FU(k, m), bits u through v of FU(k, m+1), and so on.

1.2.9.5 Cost of change from standard state: property p_5

The total cost of change from V(k) to V(i) is a weighted function of costs in time, hardware, and other pertinent parameters.

Similarly the cost of changing V(i) to V(k) is expressed in terms of the costs of time, hardware, etc.

$$V(i;p_5) = CS(k \rightarrow i) + CS(i \rightarrow k)$$

$CS(k \rightarrow i)$ is the total cost of change from V(k) to V(i)

$CS(i \rightarrow k)$ is total cost from V(i) to V(k) and both are expressed in terms of more detailed costs:

$$CS(k-i) = u_1 \; CST(k-i) + u_2 \; CSH(k-i) + \ldots$$
$$CS(i-k) = v_1 \; CST(i-k) + v_2 \; CSH(i-k) + \ldots$$

where CST(k-i) is the total cost of time, CSH(k-i) is the total cost in hardware, u_i's and v_j's are weighting factors.

1.2.9.6 Computational history of V(i): p_6

The computational history of a structure V(i) includes: 1) the total time V has been used in this structure. Various breakdowns of the time into computation time, restructuring time, and troubleshooting time; 2) comments as to the efficiency of the structure; 3) recommendations for change; and 4) list of users.

1.2.10 Notation for Micro-Description

A micro-description of V(i) expresses the properties defined in the macro-description in terms of more elementary units.

1.2.10.1 Computational characteristics, property p_1

$f(k, c; q_i) = \left(\left\{ f(j, c; q_m) \right\}, \left\{ ma(j, k) \right\}, \left\{ mi(j, k) \right\} \right)$

where $ma(j, c; q_k)$ is an elementary compound command, and $mi(j, c; q_m)$ represents elementary commands of the type $mx(j, k)$, $mz(j, k)$, $mt(j, k)$, and $me(j, k)$.

$Tma\text{-}f(k, c; q_i) = g \left(Tma\text{-}f(j, c; q_k),\ Tma\text{-}ma(j, k),\ Tma\text{-}mi(j, k) \right)$

where Tma-ma(j, k) and Tma-mi(j, k) are maximum execution times of the compound elementary and elementary commands, respectively, and

$Tma\text{-}ma(j, k) = g\left(Tma\text{-}mi(j, k) \right)$

$Tma\text{-}mi\,(j, k) = g\left(Tma\text{-}FU(j, p_k) \right)$

where $Tma\text{-}FU(j; p_k)$ is the maximum operation time of the functional unit $FU(j; p_k)$.

$Tma\text{-}FU(j; p_k) = g \left(\left\{ Tma\text{-}AM(j, k; p_m) \right\}, \left\{ Tma\text{-}LM(i, j; p_m) \right\} \right)$

where $Tma\text{-}AM(j, k; p_k)$ and $Tma\text{-}LM(j, k; p_m)$ stand for maximum operation times of amplifier, and logic basic modules, respectively.

$Tav\text{-}f(k, c; q_i)$ and $Tmi\text{-}f(k, c; q_i)$ are expressed in the same manner as given for $Tma\text{-}f(k, c; q_i)$.

$Ie\text{-}f(k, c; q_i) = (Ne\text{-}AM,\ Ne\text{-}LM)$

where Ne-AM and Ne-LM are the numbers of different basic modules used exclusively by the compound command $f(k, c; q_i)$. A listing of these may also be presented.

$Is\text{-}f(k, c; q_i) = (Ns\text{-}AM,\ Ns\text{-}LM)$

where Ns terms are defined as numbers of different basic modules shared with other compound commands. A listing of the modules and compound commands which they share may be included in this description.

1.2.10.2 Description of inventory: property p_2

A general functional unit, $FU(j;p_i)$, is described in terms of circuit modules:

$$FU(j;p_i) = \left(\left\{ AM(F, j, k; r_m) \right\}, \left\{ LM(F, j, k; r_m) \right\} \right)$$

$AM(F, j, k; r_m)$ designates the kth basic amplifier module belonging to the FU(j). Property r_m specifies the type of the amplifier module, its fan-in and fan-out capabilities, power requirements, and speed.

$LM(F, j, k; r_m)$ is the mth basic logical module of FU(j). The property r_m specifies the logical functions.

1.2.10.3 Logical and Physical Interconnections: property p_3

Logical and physical interconnections of basic modules are given in list form. The inputs and outputs of basic modules are designated by letters and numbers. The inputs are designated by letter "a", outputs by letter "z". Thus, for general basic module BM(F, j, k)

$$\left(z2\text{-}BM(F, j, k) \right) \rightarrow \left(a5\text{-}BM(F, j, m) \right)$$

denotes a logical interconnection between the output z2 of BM(F, j, k) and input a5 of the BM(F, j, m).

In order to introduce physical wiring, notation for connectors, wire types, and wire length has to be introduced. Connectors can be specified as $CO(i, j, k; s_m)$, where three coordinates are allowed for specifying the connector, and property s_m specifies the type of the connector. Wiring can be denoted as $WI(u_i, v_j)$, where property u_i specifies the kind of wiring (e.g., printed wire, twisted pair, coax, etc., and property v_i the length of the wire).

A complete notation for combined logical and physical interconnection is then as follows:

$$\left(z2\text{-}BM(F, j, k) \right) \rightarrow \left(a5\text{-}BM(F, j) \right) : WI(u_i, v_j) \rightarrow CO(i, j, k, s_m) \rightarrow$$
$$WI(u_i, v_j) \rightarrow \ldots \rightarrow WI(u_i, v_j) \rightarrow CO(i, j, k; s_m) \rightarrow WI(u_i, v_j)$$

where the physical connection is given by a chain of wires and connectors.

The coordinate system used to specify the locations of basic modules depends on the physical construction scheme to be used. A preliminary assumption of the system, however, is used to indicate the nature of location coordinates. Assuming that the system consists of frames, motherboards and modules, the location of a module is given by frame number, two coordinates for the board in a frame, and two coordinates for the module location on the board, then

$$L\text{-}BM(F, j, k) = (f; g, h; p, q)$$

where f is the frame number, g, h the board coordinates, and p, q the module position coordinates.

Assignment of locations to basic modules is not a trivial matter. At high circuit speeds, distances between modules become significant both in contributing to propagation delay and in capacitive loading of the outputs. Thus policies such as minimizing the total wire length of interconnections, or minimizing wire lengths of the interconnections with most activity may be used. Since, most likely, more sophisticated wiring (such as coax cables) is required for interconnections exceeding a certain length, the first policy may also reduce the wiring costs.

1.2.10.4 Change from standard state: property p_4

The change from standard state may be described by listing all locations where basic modules are to be replaced by different basic modules, and by listing all interconnections to be removed and all new interconnections to be established.

1.2.10.5 Cost of change: property p_5

The cost terms given in the macro-description are expressed as functions of more elementary cost terms: cost of design of the new structure, cost of wiring, cost of debugging, costs of different basic modules and connectors, costs of preparing printed circuit boards (if required), and so on. Exact nature of the cost function can be determined only after a physical construction scheme and a mechanical restructuring procedure have been adopted.

1.2.11 Change in V Structures

As stated previously, changes in V structure may be effected on several levels: electronically, electro-mechanically, and mechanically.

Electronic structure change implies enabling sets of gates, mainly in the control hierarchy, which permit a different computation to be carried out. Thus electronic switching from one structure to another is under the control of a signal, generated at some level of the control hierarchy, which can be classified as a compound command. Electronic structure change is thus a structure change in a trivial sense which, in principle, is not different from selecting one command or another in a conventional machine.

Electromechanical structure change may take the form of closing relay contacts (in which case, except for the switching time, in concept it is really equivalent to electronic switching, since construction of the structures which are switched had been done beforehand and the control signal which performs the switching, is an equivalent compound command).

It is the capability for mechanical restructuring such as physical altering of the logical properties of basic modules, altering their inter-connections, and changing their physical locations, which permits two V configurations to be completely unrelated and yet share, to a large extent, the same elementary hardware. In order to be significant, such structure changes must be effected with a minimum of V downtime (it is important to note, that during restructuring of V, F still may do useful work), both in actual restructuring and in debugging. This requirement implies maximum possible restructuring and debugging off-machine, easy changes in the logical properties of the circuit modules, and systematic procedure for changing of their interconnections. (In the extreme one can visualize completely built and debugged wiring frames for the new structure into which the circuit modules, with proper changes in pluggable cap connections, are inserted and then the new frames replace the old ones in the F+V system.)

It is clear that the physical construction scheme of V largely determines the cost of mechanical restructuring both in time and in hardware. A clear definition of such a construction scheme has to be given before realistic cost figures (which are of great importance in allocating computations to V), can be obtained.

Further, in reference to allocation of computations in the F+V system, the extent of mechanical structure change associated with a problem brings in the problem of scheduling computations in F+V, such that the total cost of mechanical structure changes for a batch of problems is minimized.

1.2.12 The Supervisory Control Unit, SC

The principal function of the supervisory control unit is to coordinate the computational activity of the F and the V parts of the F+V system. In particular, the SC must enforce the precedence requirements of computations according to the computational structure of the computational task on hand. In order to execute its functions, SC can halt and release the computations in F and in V, order information transfers between F, V, and other units of the system, and order structure changes in V.

The control sequence in SC may be a wired program, a stored program in SC proper, or a stored program in F. Communication with F and with the stored master program may be in the form of responses to interrogation commands in the stored program, e.g., at certain points the stored program will enter into interrogation loops from which it will be released only after a certain condition in SC is satisfied. Completion of operations in F are reported to the SC by special commands.

Communication with V may be on more direct basis as the SC may be viewed as the control unit on the highest control level. As such, SC can be built in the same manner as the rest of the control units in V, and all computations in V are available to it as compound commands. Completion signals are then received in the manner outlined for the control model in Section 1.2.4.

Information transfers between F and V are ordered by SC for large blocks of data under the control of its program (wired or stored) or automatically. The latter case is common for executing a computation on an argument supplied by F and transferring the result back to F. In data transfer operations, SC will perform similarly to the control unit of an F data channel, except that it will have the highest priority. In this manner, it may be possible, under certain conditions in F, to effect data transfers between F and V without actually interrupting the computations in the former.

REFERENCES TO CHAPTER I - SECTION I-2

1. Estrin, G., "Organization of Computer Systems--The Fixed Plus Variable Structure Computer", Proceedings of the Western Computer Conference, May 3-5, 1960, San Francisco, Calif.

2. Project Staff, "Annual Summary Report of Investigations in Digital Technology Research, June 1, 1959 - May 31, 1960", Report No. 60-82, Department of Engineering, University of California, Los Angeles, Calif., Nov. 1960.

3. Estrin, G., and C.R. Viswanathan, "Organization of a Fixed-Plus-Variable Structure Computer for Computation of Eigenvalues and Eigenvectors of Real Symmetric Matrices", J.ACM, Vol. 9, No. 1, January, 1962.

4. Cantor, D.G., G. Estrin, A. Fraenkel, and R. Turn, "A Very High Speed Number Sieve", Mathematics of Computation, Vol. 16, No. 78, April, 1962.

5. Cantor, D.G., G. Estrin, and R. Turn, "Computation of Elementary Functions, in x and exp x, on a Variable Structure Computer" to appear in IRE Transactions, PGEC.

6. Aoki, M., G. Estrin, and T. Tang, "Parallelism in Computer Organization--Random Number Generation in the Fixed Plus Variable Structure Computer", Proceedings of the Western Joint Computer Conference, May 9-11, 1961, Los Angeles, Calif.

7. Aoki, M., and G. Estrin, "The Fixed-Plus-Variable Computer System in Dynamic Programming Formulation of Control System Optimization Problems, Part I", Report No. 60-66, Department of Engineering, University of California, Los Angeles, May 1961.

8. Project Staff, "Annual Summary Report of Investigations in Digital Technology Research, June 1, 1960 - May 31, 1961", Report No. 62-16, Department of Engineering, University of California, Los Angeles.

9. Bussell, B., "Properties of a Variable Structure Computer System in the Solution of Parabolic Partial Differential Equations", Ph.D. Thesis University of California, Los Angeles, Calif., August, 1962.

10. Reference Manual, IBM 7090 Data Processing System, Form A22-6528-3, International Business Machines Corporation, New York, N.Y., August, 1961.

11. Muller, D.E., and W.S. Bartky, "A Theory of Asynchronous Circuits, I and II", Report Nos. 75, November 1956 and 78, March, 1957, Digital Computer Lab., University of Illinois, Urbana, Illinois.

12. Kinney, M.S., "Asynchronous Sequential Logic Design, A Selected Bibliography", Report EM-6317, Autonetics Division of North American Aviation Inc., Los Angeles, California, October, 1960.

13. Turn, R., "Assignment of Inventory of a Variable Structure Computer", Report No. 63-5, Department of Engineering, University of California, Los Angeles, January 1963.

I-3 Two-Dimensional Heat Conduction Problem

A. Problem Statement

A mathematical statement of the linear two-dimensional diffusion equation on a rectangular plate with Dirichlet conditions.

B. The Alternating-Direction Algorithm

A statement of the A-D algorithm with a derivation of the procedure for its use in computation.

C. Numerical Stability and Convergence

Development of the truncation errors involved in the A-D algorithm and an analysis leading to conditions under which the linearity requirements of a problem may be removed.

D. Computational Experiments

Comparison is made between the results of the nonlinear extension of the A-D algorithm and results of another computational procedure for the solution of a nonlinear diffusion equation.

E. Conclusions

An extension to the AD algorithm was developed. This development allows an alteration in Δt at each time step. More than this, it prescribes a computational procedure for automatically selecting the largest possible time step consistent with numerical stability. This procedure was extended to allow the inclusion of a nonlinear diffusion coefficient in the two-dimensional diffusion equation. Without a statement of this stability bound and the consequent ability to adjust Δt at each time step, the AD algorithm would lose much of its effectiveness as a method for solving the nonlinear diffusion equation.

The method was applied to a nonlinear two-dimensional diffusion equation satisfied on a square cross-section. An iterative procedure using this method yielded a transient solution which was within 3 percent of the "true" solution. In arriving at this solution, the time steps were automatically increased consistent with numerical stability.

I-4 Pattern Recognitions

INTRODUCTION

1.4.1 Pattern Recognition by Machine

We have entered an era of transition in which electronic computing machines are taking over many tasks formerly done exclusively by humans. When it is known beforehand that data is to be machine processed it is usually simplest and most economical to record the data originally in machine form. For all past history and for information whose processing cannot be well predicted it is not reasonable to expect such preparation. As a result one often finds situations in which data presented in a form suitable for humans must be processed by a machine instead. At present, with few exceptions, this data must be "coded", i.e., converted to a form acceptable by the machine in question before it can be processed. Usually this coding process involves operations such as punching holes in cards or tape and must be accomplished by humans.

In many cases, the amount of labor necessary for coding data is very small compared to the labor saved by using a machine to process the data. Sometimes, however, the time and expense of coding may negate the advantages of using a machine in the first place. Language translation, processing of bank checks, and sorting of mail are but a few of the many tasks in which the necessity of manual coding would obviously render the use of machines unfeasible. It is evident that an automatic coder, capable of rapidly and accurately reducing printed, written, or spoken data to a form amenable to machine processing would greatly extend the range of tasks to which computers can be profitably applied.

Hence it is not at all surprising that the problem of automatic recognition of visual patterns by machine is receiving a great deal of attention by research workers at present. Since numerical data and most languages are expressed by means of a small number of discrete symbols or "characters", it is clear that automatic coding can be achieved only by a machine capable of recognizing such characters. "Recognizing" in this context means the ability to respond to each character with a unique signal. In other words, a machine designed to recognize K distinct characters $P_1, P_2, P_3, \ldots, P_K$ must satisfy the following requirements:

(a) The machine must have K distinct output signals $0_1, 0_2, \ldots, 0_K$ corresponding to the characters $P_1, P_2, \ldots, P_K$ respectively, and a "reject" output 0_{K+1} such that

(b) Whenever the input to the machine is P_j $(1 \leq j \leq k)$, the output is 0_j.

(c) Whenever the input cannot be uniquely associated with one of the characters $P_1, P_2, \ldots, P_K$, the output is 0_{j+1}.

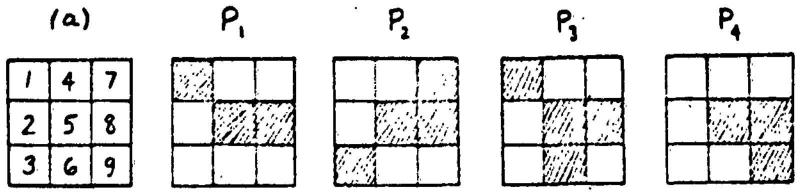
In a character recognition problem, one may begin by considering a set of ideal forms or "prototype" patterns corresponding to the characters to be recognized. Unfortunately, in actual situations the characters to be recognized may vary considerably from the prototype patterns. These variations arise in general from two sources: First, one may encounter systematic variations such as the use of type fonts which vary from the prototype font as misalignment of the characters and the reading element. Second, there are unavoidable random variations or noise which arise from such sources as over-inking of type, over-used typewriter ribbons, ink smears, etc. Noise may also be introduced by imperfections in the recognition apparatus itself. A character-recognition system must be designed to recognize not only the prototype patterns, but also all distortions and variations of them likely to be encountered. Obviously the optimal character recognition scheme for a given application will depend upon the nature of the prototype patterns, the actual characters to be recognized, and the level of accuracy which must be maintained. In Section 1.5.2, a number of character recognition schemes suitable for mechanization are outlined. These range from conceptually trivial schemes which suffice when the unknown characters are sufficiently similar to their prototypes, to more sophisticated and general methods which must be employed when dealing with, for example, sloppy handwritten characters.

1.4.2 The Concept of "Significant Features"

Character recognition schemes all make use (implicitly, at least) of certain significant features which serve to distinguish the various characters from one another. In general, these significant features occur in association with irrelevant and redundant features which do not aid the recognition process, and in fact, may hinder it. As a result many of the recognition schemes which have been proposed are grossly inefficient because this useless information is processed together with the significant information instead of being eliminated or ignored. A systematic procedure for finding the most significant differentiating features of a set of characters would therefore be a very useful aid in increasing the efficiency of pattern recognition schemes. As will be shown later, such a procedure might also be useful in a self-adaptive character recognition system (i.e., a recognition system which automatically designs or organizes itself). A successful system of this type would be an interesting advance in the field of "artificial intelligence". In this report, a quantitative measure of the "significance" of a feature is proposed and tested by a series of experiments on the IBM 7090 digital computer at Western Data Processing Center. The methods introduced in this report are based upon fundamental concepts of statistics and information theory and introduce no restrictive assumptions of an environment in which there is no noise and distortion.

In order to clarify the notion of "significance" of a feature, a simple example will now be discussed. Consider the idealized set of patterns shown in Figure 1.10. These patterns are each composed of 9 elementary areas or cells, numbered as in Fig. 1.10a. Let us restrict our attention to the 9 features F_j (j=1, 2, ..., 9) defined as follows: a pattern will be said to possess feature F_j if and only if cell C_j is black for that pattern. Thus P_1 and P_3 possess F_1, whereas P_2 and P_4 do not.

FIGURE 1.10



We observe that certain cells (2,4) are always white no matter which pattern is considered. Similarly some cells are always black (5,8). Therefore, these cells cannot serve to differentiate between any of the patterns. In other words, the features F_2, F_4, F_5, and F_8 are completely insignificant with respect to recognition of the patterns presented here.

On the other hand, observation of other cells may yield a great deal of information concerning the identity of the pattern. For example, if a pattern possesses the feature F_1, then we can conclude that it is either P_1 or P_3, and that it cannot be P_2 or P_4. If a pattern has feature F_9, it must be P_4; if it has F_6, it must be P_3, etc.

Assuming that the patterns occur with equal frequencies, we can attempt to rank the properties F_1, F_2, ..., F_9 according to significance as follows: The "don't care" features F_2, F_4, F_5, F_8 must obviously be placed at the bottom of the list. The properties F_3, F_6, F_9, and F_7 are of equal significance since they are each possessed by three patterns, and not possessed by one pattern. So, on the average, the cells (3,6,9,7) all contain equal amounts of information concerning the pattern to be recognized, by virtue of their identical distribution of black and white. Finally, we are left with F_1. Cell 1 is black for two patterns and white for the other two. It is not clear intuitively whether to rank F_1 above or below F_3, F_6, F_9, and F_7; however, it will be demonstrated later that it is proper to rank F_1 above the others. Thus, our final ranking must be as follows:

$$\begin{array}{cccc} & & F_1 & \\ F_3 & F_6 & F_9 & F_7 \\ F_2 & F_4 & F_5 & F_8 \end{array}$$

In this example we have considered a set of particularly simple properties, namely the presence of black in specified areas of the pattern field. For the purposes of this report, viz., the development of a criterion for the significance of properties, it will be advantageous to restrict our attention to these particular properties. Moreover, as a more convenient alternate viewpoint, we may regard the ranking of the features F_1, F_2, ..., F_9 as a ranking of the cells C_1, C_2, ..., C_9 according to significance. Thus we obtain the significance ranking

$$\begin{array}{cccc} & & C_1 & \\ C_3 & C_6 & C_9 & C_7 \\ C_2 & C_4 & C_5 & C_8 \end{array}$$

for the cells of the pattern matrix. Instead of the significance of general properties, in other words, we will study the significance of cells or elemental areas of the pattern field. In this manner, the experimental work will be considerably simplified, since there will be no need to use complex property-testing procedures. It is clear, moreover, that no generality will be lost, for any techniques developed for determining the significance of pattern matrix cells would also be suitable for determining the significance of properties in general.

A few salient points concerning the significance ranking demonstrated above are worth noting. First of all, the significance of a feature is a relative quantity which depends upon the observer's previous knowledge concerning the identity or properties of the unknown character. The significance ranking given above was based upon a tacit assumption of complete ignorance concerning the identity of the unknown character except that they were members of a set of four basic patterns. Suppose, however, that it is already known that the unknown pattern is either P_1 or P_3 (this situation would result if, for example, cell 1 were examined and found to be black). A glance at Fig. 1.10 reveals that under these circumstances, there would be no point in examining cells 3 and 9, since one could predict with certainty that these cells would be white. In other words, the previously significant features F_3 and F_9 have become <u>insignificant</u> as a result of the information which has been acquired*. On the other hand, we observe that F_6 is now of greater importance than before. Indeed, observation of cell 6 is now sufficient to determine the identity of the unknown pattern.

Another important point is that the significance of a feature is a function of the frequency distribution of the patterns. Suppose, for example, that P_3 is assumed to occur with a probability of say 1/100 instead of 1/4. Then a correspondingly smaller significance must be initially attributed to the feature F_6, which serves to distinguish P_3 from the other patterns.

* Note that this statement is completely analagous to the one which led to placement of F_2, F_4, F_5, F_8 at the bottom of the significance list.

A final consideration concerns the effects of noise and distortion on the significance of features. Clearly, if a particular feature is subject to a great deal of noise which occurs independently of the identity of the unknown character, its significance is much lower than it would be if the feature were noise-free and constant for any given pattern. As an extreme example, one may consider a feature so overwhelmed by noise that its presence or absence is almost independent of the identity of the unknown pattern. Obviously, such a feature is rendered insignificant by the noise.

On the other hand, if the noise in a particular cell is not independent of the character being examined, the noise may actually convey information concerning the identity of the unknown character, and the significance of the cell might therefore be increased. For example, consider again the ideal pattern set for Fig.1.10, and assume that the patterns are affected by noise in cell 2 as follows: (a) When the unknown pattern is P_1, cell 2 appears black instead of white with a probability of 1/2. (b) If the unknown pattern is not P_1, cell 2 always appears white as in the ideal case. Suppose now that an unknown pattern is examined, and it happens that cell 2 is black. One could immediately conclude that the unknown pattern is P_1. This indicates that cell 2 has a certain amount of significance. In the ideal case, on the other hand, cell 2 is always white and hence has no significance whatsoever. Therefore we conclude that noise of the type considered increases the significance of cell 2.

Any generally useful quantitative measure of the significance of features must deal with all of the above considerations. The effects of noise, variation of the probabilities of occurrence of the patterns, and partial knowledge concerning the identity of the unknown pattern, must all be properly taken into account. A number of schemes for determining significant properties (or significant cells, to be more precise) have been proposed. The writer will attempt to establish that the significance criterion to be presented in this report is quite generally valid when all the above factors are operative. Previously described methods are of relatively restricted validity and may be utilized successfully to the extent that the pattern set is constrained by the same restrictions. Several such methods will be discussed in Section 1.5.5, where a review of recent literature pertaining to the significance of pattern features (or areas) is given.

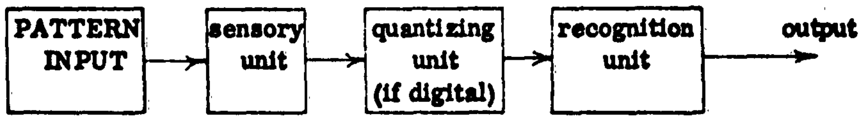
I-5 General Discussion and Review of Automatic Pattern Recognition

The purpose of this section is to acquaint the reader with the basic principles of automatic character recognition devices and to review recent articles and theses dealing with problems related to those considered in this work.

1.5.1 The Fundamental Components of Pattern Recognition Systems

We will begin by discussing the fundamental components of pattern recognition machines. These are indicated in the block diagram of Figure 1.11. It should be pointed out that the block diagram is an idealization, and that in actual systems it is not usually possible to separate the machine into independent units performing separate functions as indicated. Nevertheless, the breakdown according to functions as shown is useful conceptually and will enable us to discuss pattern recognition in general terms. The pattern input block shown represents the pattern itself, the document on which it appears, the feeding and alignment mechanisms, etc.

FIGURE 1.11



In visual character recognition, the function of the sensory unit is to convert the character, which is a pattern of light and dark areas, into electrical or other signals which can be conveniently processed by the machine. In pattern recognition machines, the sensory unit generally involves an arrangement of lenses, light sources, and photocells. In special systems employing characters written in magnetic ink, the sensory device is a magnetic read head.

In many practical systems, the sensory unit incorporates some type of scanning mechanism. The scanning mechanism consists of a small sensory element which sequentially inspects various areas of the pattern. The scanning path employed may either be predetermined, or it may be made to depend upon the pattern being scanned (e.g., the scanner might be designed to follow the boundary of the pattern presented for identification). Most commonly, the scanner follows a continuous, predetermined path covering the entire pattern area.

The quantizing unit is an essential part of digital pattern recognizing machines. In this unit, the output of the sensory devices, which is generally a continuous function of space and/or time coordinates, is quantized for digital processing. The most common type of quantization is shown in Figure 1.12.* The pattern is superimposed on a Cartesian grid or matrix, and each cell is considered to be either all black or all white, according as the amount of black area within the cell exceeds or fails to exceed a given threshold.

Coding is a process whereby the quantized character signal is converted into a series of numbers or code. An extremely simple coding method for the quantized character shown in Fig. 1.12 consists of forming a matrix of binary digits by placing a "1" in all black cells, and a "0" in all of the white

* See page 80.

cells. (Fig. 1.13). Thus each pattern is represented as a matrix whose elements are ones or zeros.

It is useful to distinguish between "relative" and "absolute" codes.* A relative code for a group of patterns is a code which contains just enough information so that the representation for each pattern is distinct. An "absolute" code, on the other hand, is one which contains essentially all the information in the pattern. Given an absolute code representation for a pattern, it is possible to reconstruct the original pattern with an accuracy limited only by the "coarseness" of the quantization process. The binary digit matrix of Fig. 1.13 is an example of absolute coding. Obviously, for the purposes of character recognition, a relative code is sufficient. Indeed, since a relative code usually yields shorter representations for each pattern than an absolute code, use of a relative code may result in considerable simplification of the succeeding recognition unit.

The recognition unit operates upon the output of the coding unit (in digital systems), and determines the identity of the unknown pattern. In analog systems, the recognition unit generally processes the output of the sensory block directly.

Another operation which is often necessary in pattern recognition systems will be termed "pattern processing". The purpose of pattern processing is to aid recognition by altering the unknown pattern, (or its coded representa-

* This terminology is due to Kharkevich (15)

tion), so that it will more closely resemble the corresponding prototype or ideal pattern. Pattern processing involves operations such as pattern centering, removal of spurious ink spots, smoothing of jagged boundaries, etc. Operations of this type are usually necessary when the patterns dealt with are subject to significant amounts of distortion and noise. A "pattern processing" unit was not included in the block diagram of Fig. 1.11 because this operation cannot be localized at all. Pattern processing may occur in any (or all) of the blocks indicated, including the recognition unit.

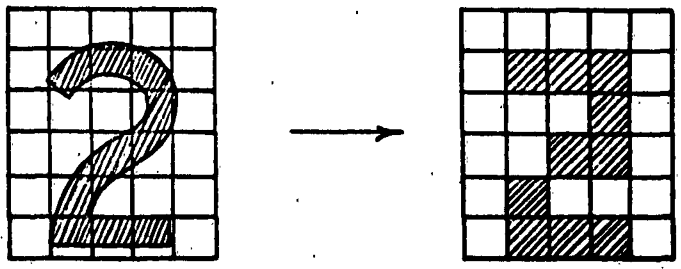
Sometimes it is possible to eliminate the need for pattern-processing by employing a code which yields representations which are invariant with respect to the types of distortion present. It is clear that if the output of the coding unit can be made independent of the distortion and noise, so that it depends only upon the identity of the input pattern, there is no need to remove the noise through separate pattern processing. A great saving in hardware may be achieved in this manner.

1.5.2 Automatic Pattern Recognition Techniques

In order to illustrate some of the ideas introduced above, the main types of automatic pattern recognition schemes will now be discussed. We will begin with a discussion of the template-matching (or "masking") technique, which is one of the simplest (but least powerful in noise and distortion tolerance) of the methods which have been proposed.

As the name implies, this technique involves producing a "template" or "mask" for each character. Each template is identical to the assumed ideal form of the corresponding character. Recognition of an unknown character is accomplished as follows: the unknown is compared with each template by simple superposition, and is subsequently identified as the character whose template gives the "best fit". More precisely, let us assume that a digital computer is being employed to carry out the template-matching process. The templates are stored in memory in the form of binary digit matrices of the type illustrated in Fig. 1.13. The unknown character is also converted to binary matrix form and compared with each template in turn by (1) counting the number of times 1's occur in corresponding cells of the unknown and the template, and (2) subtracting from this number the number of times a cell of the unknown contains a zero while the corresponding cell of the template contains a one (or vice versa). In this manner, a measure of similarity between the unknown character and each prototype character is determined.

In order for this procedure to work satisfactorily, it is clear that the unknown characters must not deviate to any significant extent from their templates. More precisely, for proper recognition an unknown character must not deviate from its corresponding template by more than half of the "similarity distance" between the template and its closest neighbor in the template space. Therefore, the unknowns must be very accurately centered, corrected for tilt, and optically (or otherwise) corrected for variations in size. Even when provision has been made for accomplishing these difficult adjustments, the method remains weak in the sense that new templates usually must be defined for different fonts. For example, if an "H" appeared in the variant form "A", it would undoubtedly be misrecognized as an "A". It is obvious that this system cannot be depended upon except for the simplest applications. More elaborate and powerful template-matching systems employ "statistical templates" which are based upon actual samples of the characters rather than on assumed ideal forms.



Pattern Quantization
FIGURE 1.12

0	0	0	0	0
0	1	1	1	0
0	0	0	1	0
0	0	1	1	0
0	1	0	0	0
0	1	1	1	0

or

```
0 0 0 0 0
0 1 1 1 0
0 0 0 1 0
0 0 1 1 0
0 1 0 0 0
0 1 1 1 0
```

Binary digit code matrix
FIGURE 1.13

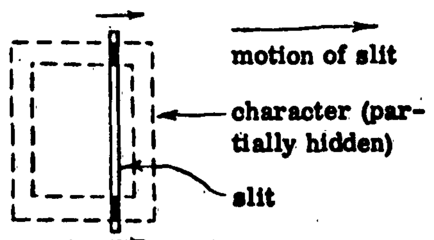


FIGURE 1.14a

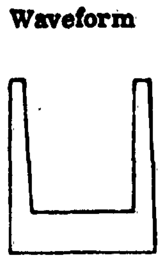


FIGURE 1.14b

An important class of pattern recognizers employs the <u>single-slit scanning procedure</u> (5). The character is scanned by a thin slit which moves across it as shown in Fig. 1.14a. The relative motion of the slit is perpendicular to its length. The result of such scanning is an analog voltage waveform characteristic of the figure being scanned. This waveform is then compared with signals representing each of the possible characters. The compare-character signal with the best match to the scanned signal is selected as the machine-read character. With the slit-scan technique, information about the location of ink along the length of the slit is lost. For this reason, the discriminating power of recognition systems using this type of scanning is somewhat restricted. In general, the number of characters must be kept small, and the font must be specially designed to insure that the signals produced by each character are sufficiently different from one another (8). These drawbacks are to some extent compensated for by the greater speed and positioning tolerance achievable in comparison with systems employing two-dimensional scanning.

There are two common methods of generating the analog waveform. In the first method, optical transducers are employed, and the waveform voltage at each instant is proportional to the amount of ink within the area of the slit. The result of scanning the character "0" by this method is indicated in Fig. 1.14b.

In the second method, the characters are printed in magnetic ink and the slit is replaced by a magnetic read head. The waveform voltage at each instant is then proportional to the <u>derivative</u> of the black area under the reading head. A system using this type of scanning has been perfected jointly by Stanford Research Institute and General Electric (5) and is now widely used by banks in check-processing. A specially designed type font is employed, and the printing must be held to very close tolerances. The most practical method for comparing the scanned waveform with the prototype signals is based on the concept of "matched filters". A matched filter is "matched" with respect to a given signal in such a way that the impulse response of the filter is the time-inverse of the signal. That is, h(t)=As(b-t), where h(t) is the filter impulse response, s(t) is the signal being matched, and A and b are constants. If the input to this filter is the matched signal s(t), then the output o(t)= $\int_0^t h(t-\tau)s(\tau)\,d\tau = \int_0^t As(b-t+\tau)\,s(\tau)\,d\tau$.
At the particular instant t=b, we find: $o(b) = \int_0^b A(s(\tau))^2 d\tau$. It is an immediate consequence of Schwartz's Inequality that $|o(t)| \leq |o(b)|$ for all t. Suppose now that a signal q(t)≠s(t) is applied as the input and that q(t) has been nor-

malized so that $\int_0^b q^2(t)\,dt = \int_0^b s^2(t)\,dt$. Denoting the output under these conditionsby g(t), Schwartz's Inequality yields the relation $|g(t)| \leq |o(b)|$. Assume now that matched filters have been constructed for each of the compare-character waveforms. When an unknown character is scanned, the waveform generated is applied to each matched filter f_i. The output $o_i(t)$ of each filter is normalized and monitored during an interval of time containing the "in-phase" point t=b, and the maximum value m_i of each output held. It is clear from the preceding remarks that the largest of the m_i will be produced by the matched filter corresponding to the character being scanned. In this manner, recognition is accomplished.

Matched filters can be approximated using conventional filter theory, or more simply by use of tapped delay lines as shown in Fig. 1.15. When an impulse is applied to the input terminal, a voltagepulse is initiated which travels along the line at the line's characteristic velocity.

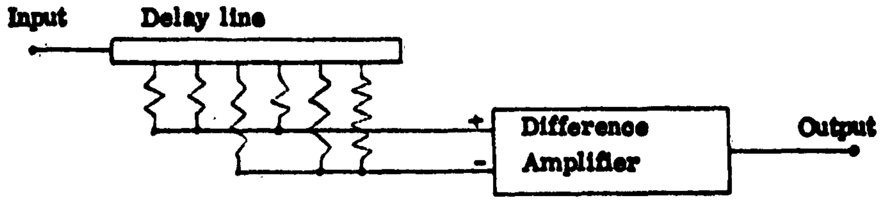


FIGURE 1.15

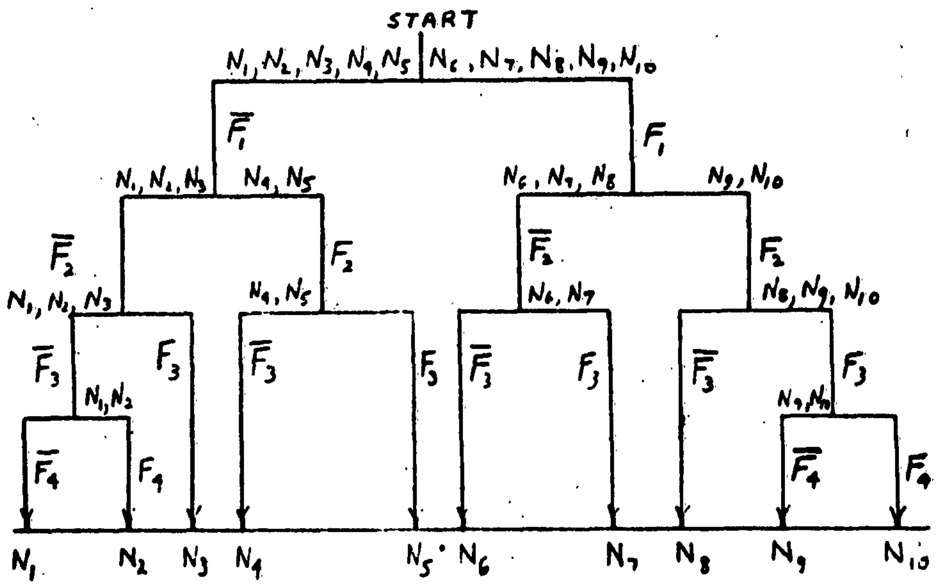
The resistors are adjusted so that the output of the system assumes the proper matched-filter values at the instants when the disturbance in the delay line reaches each tap. The difference amplifier makes it possible to simulate matched filters having both positive and negative outputs. The accuracy of the representation is dependent upon the number and closeness of the taps.

The character-recognition schemes discussed so far are obviously of restricted usefulness. A few more powerful methods will now be considered. Foremost among these in importance is the "property list" method (19). Basically, this method consists of making a list of properties characteristic of each pattern to be recognized. The unknown pattern is then tested for the presence or absence of each of the properties on each list. If the property lists are properly constructed, it will always be possible to identify the unknown pattern on the basis of the outcomes of the tests.

For any given character, it is usually possible to find some properties which will persist even when the characters are severely distorted. For example, the patterns "C", "Ͼ", "c", and "⊏", which are all common representations for the third English letter, all have the property of being "open toward the right". It is very uncommon to encounter a "C" which does <u>not</u> have this particular property, even in the case of very sloppy handwriting. Because of this "persistence of properties", the property-list method is applicable to the recognition of highly distorted characters.

Many variations and refinements of the method are possible. For example, for each pattern a list of properties <u>not</u> characteristic of the pattern may be assembled. If the unknown pattern possesses properties not possessed by a given prototype, then it can be concluded that the unknown differs from that prototype. In this system, the unknown is identified by a process of elimination.

In general, it is advantageous to combine the use of characteristic and non-characteristic features in one recognition scheme. To illustrate the general procedure, assume that 10 characters $N_1, N_2, \ldots, N_{10}$ are to be recognized, on the basis of the properties F_1, F_2, F_3, F_4. One possible recognition scheme is illustrated in the "character tree" diagram of Fig. 1.16. In this situation, we observe that characters $N_1, N_2, \ldots, N_{10}$ possess property F_1, while characters $N_6, \ldots, N_{10}$ do not; N_4, N_5, N_8, and N_9 possess F_2, while N_1, N_2, N_3, N_6 and N_7 do not, etc.



Direct Logic Recognition Scheme

FIGURE 1.16

A logical representation for the characters can be written as follows:

$$N_1 = \bar{F}_1 . \bar{F}_2 . \bar{F}_3 . \bar{F}_4 ; \quad N_2 = \bar{F}_1 . \bar{F}_2 . \bar{F}_3 . F_4$$

$$N_3 = \bar{F}_1 . \bar{F}_2 . F_3 ; \quad \ldots N_{10} = F_1 \cdot F_2 \cdot F_3 \cdot F_4$$

If desired, these formulas may be expressed in the form of 4-digit binary number codes by placing a 1 in the ith significant digit of the code, if the property F_i is possessed by the corresponding character, and a zero otherwise. In this manner we obtain the following code representation:

N_1 ---- 0000

N_2 ---- 0001

N_3 ---- 0011 or 0010

...

N_{10} ---- 1111

In the above recognition logic, it will be observed that at the ith stage in the recognition process, the unknown character is always tested for the presence of a specific property F_i, regardless of the outcome of previous tests. In more complicated systems, the property which is tested for at each stage is a function of previous test results, as in the following scheme involving 8 characters $N_1, N_2, \ldots, N_5$ and 5 properties $F_1, F_2, \ldots, F_5$.

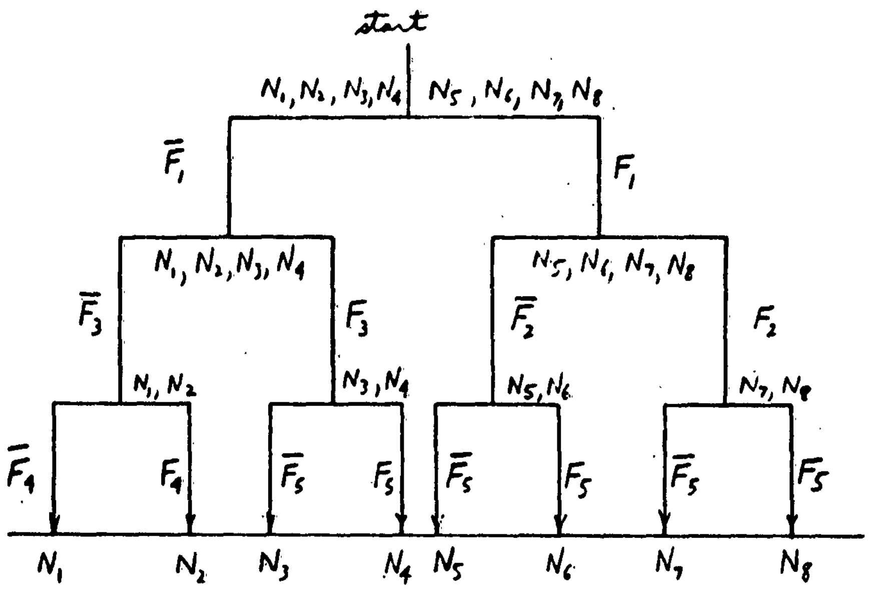


FIGURE 1.17

Three-digit binary number codes for the characters can be established as follows: the i^th^ significant digit is set equal to 1 if the property tested for at the i^th^ stage is present; otherwise, the i^th^ significant digit is made zero.

In direct logic recognition schemes such as those illustrated above, the binary code representation for each character must be distinct in order to insure correct recognition. Therefore, if the number of patterns to be recognized is P, and ℓ is the integer such that $2^{\ell-1} < P \leq 2^{\ell}$, the binary codes must be at least ℓ digits in length. This means that the recognition process must consist of a sequence of at least ℓ tests. It is convenient to write $\ell = \{\log_2 P\}$, where $\{\chi\}$ denotes the smallest integer greater than or equal to χ. If the properties to be tested for are not maximally significant, a much larger number of tests may be necessary. In order to speed up and simplify

the recognition process, it is usually desirable to minimize the number of tests necessary by employing very significant ones only. On the other hand, the most significant tests may be very inconvenient and complicated. Therefore, in practice, a compromise must be made between simplicity and significance in choosing the properties to be tested for.

If the patterns are all in binary matrix form, the simplest and most convenient property to test for is the presence (or absence) of a "1" in a particular cell of the matrix. If the resolution of the pattern matrices is sufficient (i.e., if the character field has been subdivided into a sufficient number of cells), it is always possible to recognize a set of distinct patterns by testing the series of cells in this manner. The problem of reducing the number of tests to a minimum is replaced, in such a recognition system, by the problem of minimizing the number of cells which must be tested in order to insure identification of the unknown pattern.

A serious shortcoming of direct logic recognition schemes is that they are limited by the weakness of their worst tests. In other words, if an unknown character failed (because of noise or distortion) to have one of the properties characteristic of its prototype, an error might result, even if all the other significant properties of the prototype were present. The spuriously absent property would cause the recognition process to follow the wrong branch of the character tree, unless complicated error-detecting and error-correcting loops are included in the system. For this reason, property-list recognition schemes employing direct logic can only be used when a sufficient number of simple, reliable tests can be found.

One way to get around this difficulty is to perform the tests in parallel, rather than sequentially, as in the direct logic system. No decision concerning the identity of the unknown character is made until the results of all the tests have been obtained. In the simplest system, the number of properties possessed jointly by the unknown character and each prototype are tabulated from the test results, and recognition is based upon the resulting crude similarity index. This is the simple "majority rule" technique. In more sophisticated methods, such as that used by Worthie Doyle (6) to recognize sloppy handwritten capital letters, statistical weighting factors are employed in the computation of the similarity index. The weighting factors are obtained by combining assumed a priori probabilities for the characters with the results of a "census", in which each of the tests is applied to a large representative sample of each character and the outcomes recorded.

If a sufficiently large number of independent tests are used, the probability of misrecognizing a character due to spurious test outcomes can be made small. This holds true even if the tests are not very reliable and significant. For example, Doyle selected a series of tests on the basis of their being easily programmed and fast running on the general purpose digital computer he used to test his system. He gave little thought to their reliability or their power to differentiate between the characters. Nevertheless, he achieved correct recognition rates of approximately 87% for handwritten capital letters.

One great disadvantage of most property-list recognition schemes is their relative inflexibility. Once such a system has been built to recognize a particular set of characters, it usually cannot be used to recognize a different set of characters without radical alterations. This is particularly true of direct-logic recognition systems. For applications in which it is desirable to have a single machine which can be used (with only minor adjustments, if any) to recognize arbitrary sets of characters, a machine capable of "learning" to recognize characters would be very valuable. A few such "self-adapting" systems have been built for experimental purposes. In general, the system is trained as follows: A large number of representative character samples are presented to the system. The machine is told the identity of each character as it is presented, and on the basis of this information "learns" to recognize the various characters. If the machine is to employ a property-list recognition scheme, this learning phase must involve a search for significant properties. As yet no useful machine has been developed which can do this satisfactorily, although a number of workers have studied the problem*. The property significance criterion introduced in this report might be used to advantage in a system of this type.

At present, the most successful self-adaptive systems simply subject the sample characters to an arbitrary sequence of simple, predetermined tests. Recognition of unknown characters is based on a statistical analysis of the test outcomes obtained during the learning phase. Doyle's system,

* See page 93 of O. G. Selfridge, "Pattern Recognition and Modern Computers". (22)

described above, is of this type, as are those of Baran and Breuer, which will be discussed in Section 1.5.4.

Another class of self-adapting systems, typified by the Perceptron of Rosenblatt (21), are based upon assumed models of the brain, with electronic elements simulating neurons, synapses, etc. The Perceptron model in simplified form is shown in Fig. 1.18.

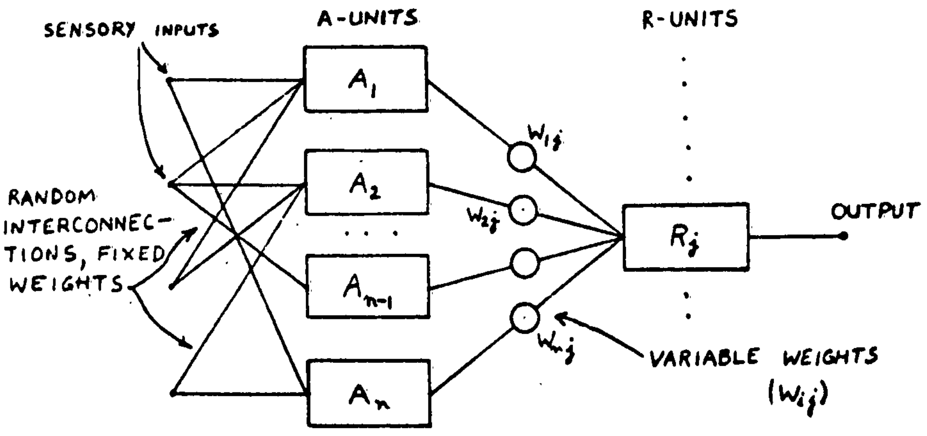


FIGURE 1.18. Perceptron Model

Sensory inputs are mapped by means of random connections onto a series of simulated neurons called A-units. Like their biological counterparts, each of the A-units is quiescent, unless the sum of the "stimuli" reaching it from the sensory elements exceeds a certain threshold. The output x_i of each A_i, weighted by a variable factor w_{ij}, is applied to each of another set of neurons called R-units (to simplify the diagram, only one R-unit is shown in Fig. 1.18). The output 0_j of R_j is determined as follows:

$$O_j = 1 \text{ if } \sum_i x_i W_{ij} \geq \theta_j$$

$$O_j = 0 \text{ if } \sum_i x_i W_{ij} < \theta_j \qquad (j = 1, 2, \ldots, m)$$

where θ_j = threshold of R_j, m = total number of R-units.

During the learning phase, the values of the W_{ij} are changed whenever the output of R_j does not correspond to some arbitrary desired response D_j for the given input pattern. If the patterns to be recognized are sufficiently distinct and sufficiently undistorted, if a sufficient number of sensory and neural elements are employed, and if a suitable method for adjusting the values of the W_{ij} is used, the system will converge to a state in which it will yield the desired responses to input patterns.

As yet, the optimal method for adjusting the W_{ij} is not known. However, a number of rules have been tried, mostly with some success. Verification of network learning has been achieved both by means of digital computer simulations and by experiments on a hardware model (4). The latter has successfully learned to recognize the letters of the English alphabet, in fixed position and font.

1.5.3 Automatic Character Recognition in Review

The purpose of this section is to review a few recent developments in the field of automatic character recognition. Comprehensive reviews of the literature through approximately August 1960 have been given by Breuer and by Baran (3,1). Therefore, only very recent articles are dealt with here. It is hoped that the systems discussed here, in addition to those referred to in other sections of this chapter, will provide the reader with sufficient background to facilitate his understanding of subsequent chapters of this report.

In the last section it was pointed out that it is highly advantageous to employ encoding schemes which are invariant with respect to the most common distortions or variations in the characters to be recognized. An encoding scheme which hasthe property of registration invariance (i.e. invariance with respect to vertical and horizontal translations of the character) is described by L. P. Horwitz and G. L. Shelton of IBM (14). Suppose the character is quantized into an NxN mosaic of black and white squares as described previously. We establish a Cartesian coordinate system on a (2N-1)x(2N-1) field so that each elemental area of the field is associated with a vector $\vec{\lambda}=(x,y)$, as in Fig. 1.19. Assuming that the NxN pattern matrix is located somewhere on the (2N-1)x(2N-1) field, we define a function $f(\vec{\lambda})$ as follows: $f(\vec{\lambda})=f(x,y)=1$ if the elemental area whose coordinates are (x,y) is black; otherwise $f(\vec{\lambda})=0$. Consider the function $D(\vec{R})=\sum_{\lambda} f(\vec{\lambda})f(\vec{\lambda}-\vec{R})$ whose domain is the (2N-1)x(2N-1) field. Evidently, $D(\vec{R})$ equals the number of pairs of ones on the pattern matrix with relative separation $\vec{R} = (\Delta x, \Delta y)$; this number is obviously invariant with respect to shifts of the NxN pattern matrix on the (2N-1)x(2N-1) field, since the relative separation of a pair of ones is registration invariant. $D(\vec{R})$ is known as the discrete autocorrelation function of the pattern from which it was derived. It is necessary to extend $f(\vec{r})$ from the NxN domain to the (2N-1)x(2N-1) matrix since the vector $(\vec{r}-\vec{R})$ does not necessarily lie on the original NxN matrix.

The discrete autocorrelation function is not in general a 1-to-1 function of the set of patterns to be recognized; indeed, for any particular pattern P, $D_P(\vec{R}) = D_{P'}(\vec{R})$, where P' is a new pattern formed by rotating P through 180°. Hence, $D(\vec{R})$ does not always form an adequate description of a set of patterns for recognition purposes -- for example, one could not distinguish a "6" from a "9" given only the discrete autocorrelation function. However, given a set of prototype patterns whose discrete autocorrelation functions are distinct, it is quite practical to base a recognition scheme upon the autocorrelation function, even in the presence of considerable noise and distortion. Horwitz and Shelton took as a measure of the "similarity" between two patterns A and B the quantity $S_{AB} = \sum_{\vec{R}} D_A(\vec{R}) D_B(\vec{R}) \Big/ \left(\sum_{\vec{R}} D_A^2(\vec{R})\right)^{1/2} \left(\sum_{\vec{R}} D_B^2(\vec{R})\right)^{1/2}$. It can be seen that S_{AB} represents the cosine of the angle between two multidimensional vectors. By Schwarz's Inequality, $0 \leq S_{AB} \leq 1$, with $S_{AB} = 1$ only if A and B are identical except for rotation through 180° or shifting. If the autocorrelation functions

$D_1(\vec{R}), D_2(\vec{R}), \ldots, D_m(\vec{R})$ of the prototype patterns $P_1, P_2, \ldots, P_m$ are known, then an unknown pattern, U, can be identified by computing its "similarity" $S_{U1}, S_{U2}, \ldots, S_{Um}$ to each of the possible prototypes and associating it with the prototype to which it is most "similar". To prove the feasibility of the recognition scheme, a small number of alphabetic characters were digitally processed by Horwitz and Shelton with encouraging results.

Work is now being done on the problem of constructing a practical registration-invariant reading machine based on the autocorrelation function. It can be shown that in the limit, as the resolution of the pattern matrix is increased, the quantity S_{AB} defined above can be found by means of specially designed optical filters. The character A to be recognized must be in the form of a negative photographic transparency. The Fraunhofer diffraction pattern produced by this negative is projected by a collimating lens through a normalized optical filter made by photographing the Fraunhofer diffraction pattern of the prototype character B. The total light energy passing through the system is then proportional to S_{AB}. In practice the character could be recognized by the arrangement shown in Fig. 1.20, in which the unknown character is compared in parallel with each of the prototypes.

At present the optical method is not practical, since the document must be in the form of a transparency. Also, the light levels inherent to the system are very low. An electronic system for generating autocorrelation functions has therefore been developed by Shelton, McDermid and Petersen.

FIGURE 1.19

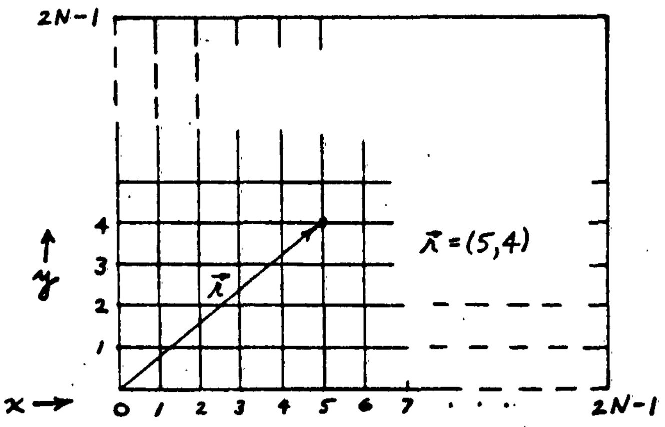
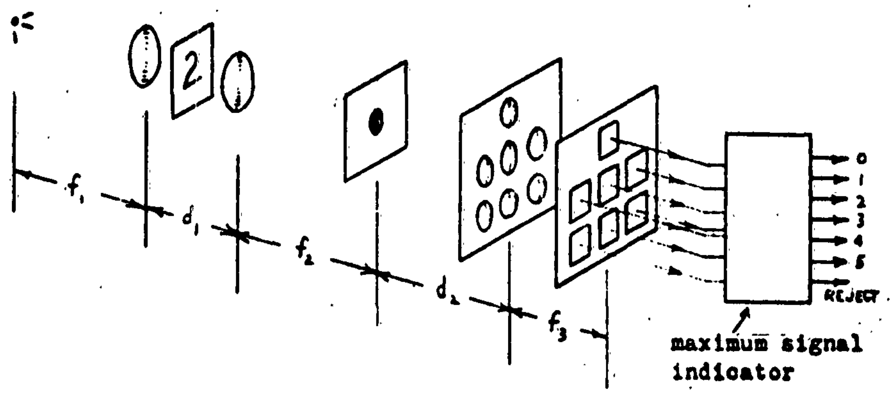


FIGURE 1.20
Parallel recognition through Fraunhofer diffraction



The unknown character, in $M\times(2N-1)$ binary matrix form, is entered, bit by bit, into a shift register as shown in Fig. 1.21. The order in which the bits enter the register is shown by the numbering in Fig. 1.22. The first position in the register is ANDED independently to all the other register positions, and the outputs of the ANDS each fed into a counter. After the matrix has completely entered the shift register, the numbers in the counters will be (reading from left to right):

(1) The total number of bits in the pattern.

(2) Number of 1's separated by 1 shift in $+Y$ direction.

(3) Number of 1's separated by 2 shifts in $+Y$ direction.

. . . .

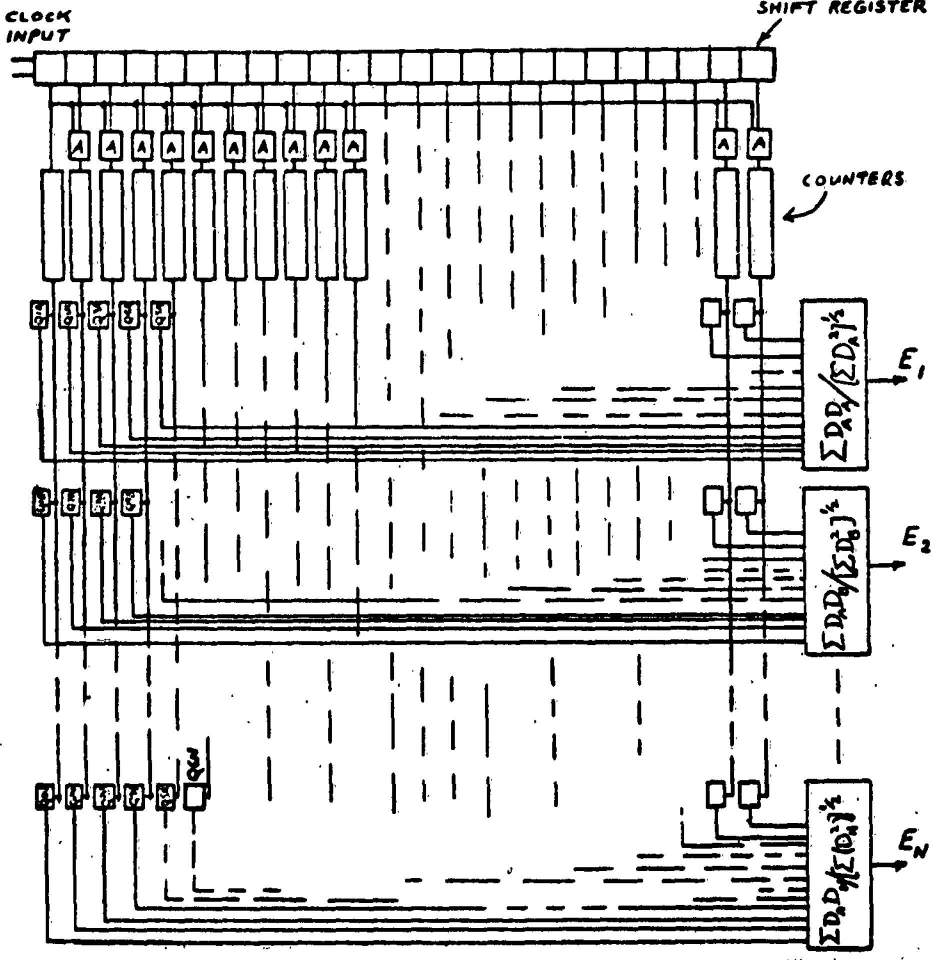
(N) Number of 1's separated by 1-unit shift in X-direction and N-1 shift in $-Y$ direction.

(N+1) Number of 1's separated by 1-unit shift in X-direction and N-2 shift in $-Y$ direction.

and so forth, every 2Nth position corresponding to an individual X-shift. The similarity function S_{AB} defined above can be computed by multiplying the numbers in the counters by the appropriate factors for each comparison channel, summing, and normalizing as shown in the block diagram of Fig. 1.21.

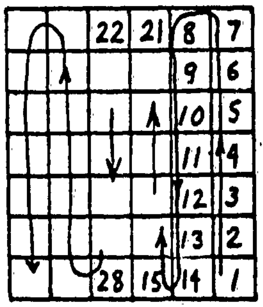
In this system, it is possible to distinguish between two characters which differ only by a rotation through 180° by looking at the partial autocorrelation functions which exist in the counters as the input enters the register. In general, at corresponding times, these will be distinct, even for this ambiguous case.

FIGURE 1.21


CLOCK INPUT
SHIFT REGISTER
COUNTERS
A
Q
$\Sigma D_A D_n / [\Sigma D_A^2]^{1/2}$
E_1
$\Sigma D_n D_B / [\Sigma D_B^2]^{1/2}$
E_2
$\Sigma D_n D_N / [\Sigma D_N^2]^{1/2}$
E_N

A="AND"; Q =Normalized reference multiplied by counter value. Omitted units are indicated by dotted lines.

FIGURE 1.22



A line-drawing pattern recognizer developed by L. D. Harmon of Bell Telephone Laboratories (13) makes use of a scanning system yielding character descriptions which are essentially invariant with respect to rotation and changes in size. At present the system is capable of recognizing line drawings of circles, triangles, squares, pentagons, and hexagons. It can also count up to six small opaque objects. With more sophisticated circuitry, it appears possible to extend the system to recognize hand-written or printed alphanumeric characters as well.

The scanning field is an arrangement of c concentric rings of r elements each (Fig. 1.23). Scanning consists of sequentially observing the elements in each ring, starting from the innermost ring and proceeding outward. Thus any pattern in the scanning field is transformed into a sequence (in time) of c binary numbers, each r digits in length. If these numbers are arranged in the form of a matrix as shown in Fig. 1.24, it is evident that changes in size of the scanned figure will result merely in vertical shifts of the pattern of ones on the matrix, as long as the figure remains on the scanning field. Similarly, rotations of the figure being scanned will result in horizontal shifts of the pattern of ones. In case a portion of the pattern is shifted off the matrix, due to a rotation of the figure on the scanning field, it will re-appear on the opposite side. Hence this type of scanning yields descriptions which are indeed quite invariant with respect to rotation and size changes.

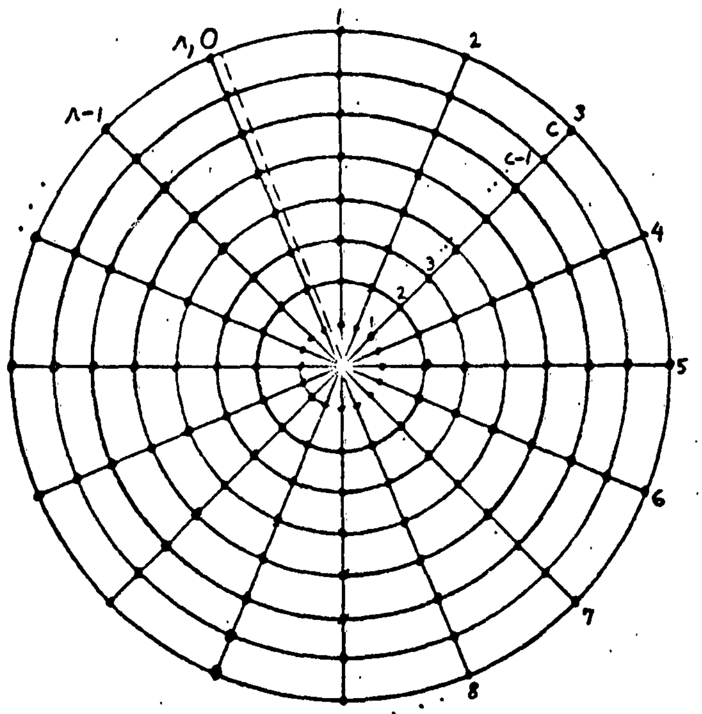


FIGURE 1.23

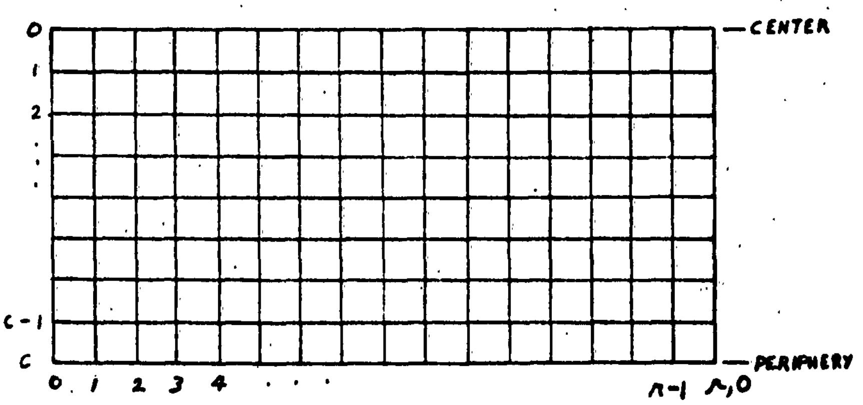


FIGURE 1.24
Matrix obtained from scanning field by cutting along dotted line in Fig. 1.23 and straightening.

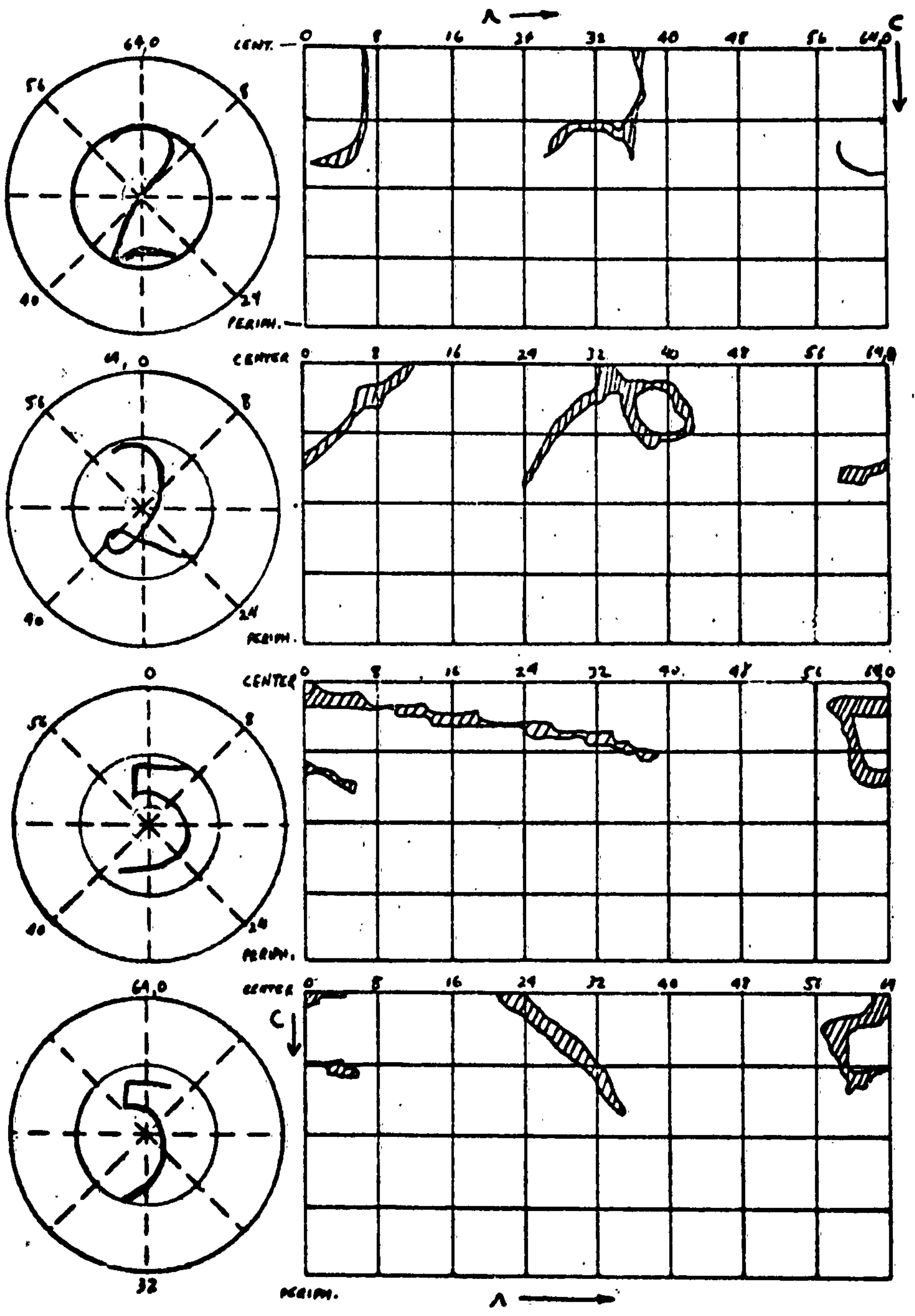


FIGURE 1.25

The results of scanning a few centered numerals are shown in Fig. 1.25. It will be observed that there exist certain properties which distinguish the patterns corresponding to various forms of the figure "2" from those due to scanning of a "5". The two valued function for $0 \leq r \leq 8$ and $60 \leq r \leq 64$ establishes the presence of the top bar and top of the curved portion of a 5, for example. This feature would be absent from a 2, which instead would probably have an ascending function for $0 \leq r \leq 8$ as shown, etc. If tilting is present, relative rather than absolute values of r̲ must be considered, of course, but this would not be a difficult task for a reading machine to perform (in fact, the registration-invariant autocorrelation function of Horwitz and Shelton might be employed at this point). In principle, at least, it would seem that the scanning system considered here could form the basis of a useful direct or majority-logic recognizer for characters subject to tilting and variations in size.

The simple polygon-recognizer and object counter actually built by Harmon and his associates, employs a mechanically puckered ring of 32 photocells as a scanning mechanism. Whenever a particular photocell encounters a black area, this fact is recorded in associated circuitry. If we associate the value 0 with all photocells which have not encountered a black area, and the value 1 with those which have, it is evident that counting a number of objects having sufficient angular separation is equivalent to counting a series of strings of 0's and 1's (Fig. 1.26). Recognition of polygons is based on the observation that as the scanning ring intersects a vertex, two strings of l's must flow together (Fig. 1.27). By counting the number of times this occurs during the scanning cycle, a convex polygon or circle can be identified. All counting operations mentioned here are performed by an arrangement of thyratrons, neon bulbs, and photocells.

A pattern-recognition system bearing certain similarities to Harmon's has been designed by J. R. Singer of the University of California at Berkeley (23). Singer's scheme employs an interesting image-sensing unit which is based upon a model of the human retina and optic nerve. This sensory unit is composed of a group of basic contour-sensing units, one of which is shown in Fig. 1.28 . It will be observed that an output will be produced unless all four of the photocells a, b, c, and d are equally illuminated. In other words, there will be an output from the logical unit if there is any variation in light intensity on the field scanned by the unit. By similarly differentiating the outputs of four of these basic logic units, higher-order outputs are obtained; by repeating this process, extremely complex and discriminating sensory systoms can be produced. The output lines of the differentiating circuits at various levels are termed "signal conductors" or "optic fibers". Singer has adopted a sensory system consisting of 72 photosensitive elements and 36 optic fibers arranged in a circular configuration (Fig. 1.29).

The optic fibers are connected to a device called a "delay transformer" which transforms the image from "optic fiber space" into a "delay space". Essentially, "delay space" is a space in which the image retains its form,

FIGURE 1.26

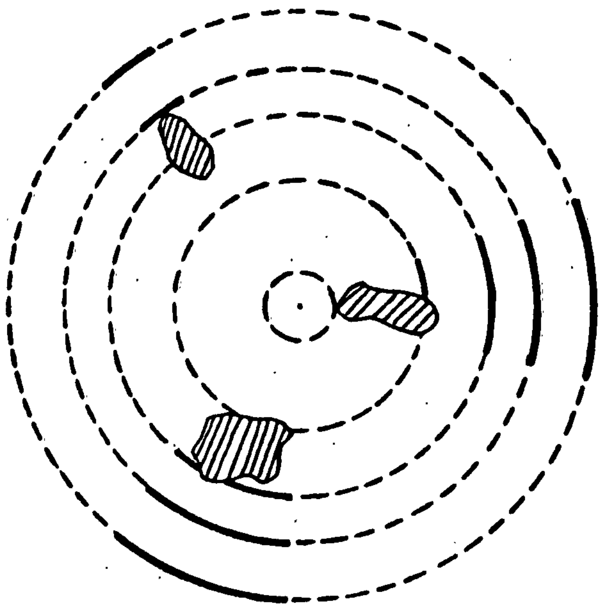
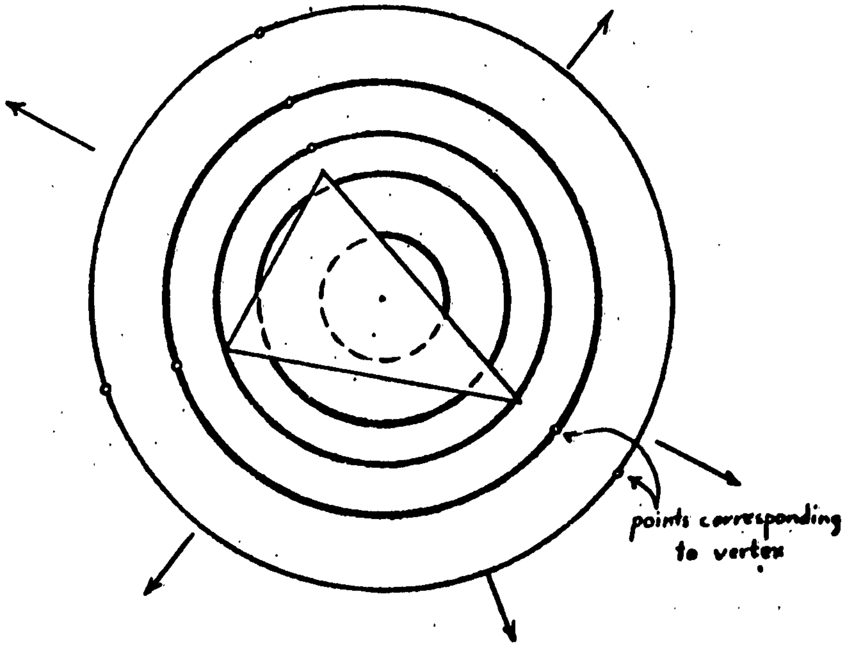


FIGURE 1.27
Recognition of triangle through vertex detection

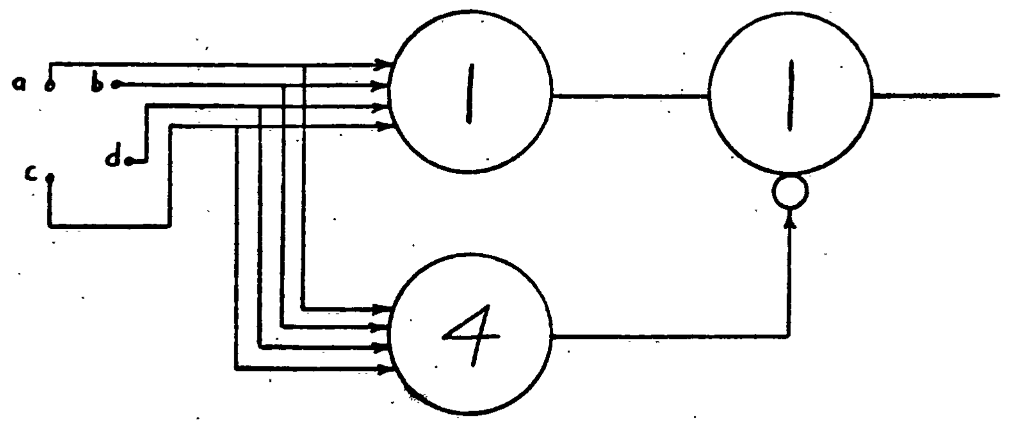


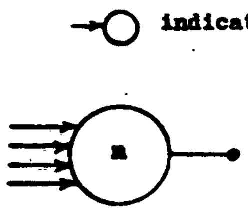
but expands in size at a rate which is a function of the time. A simple delay transformer might be constructed by connecting each optic fiber through a delay line or through a series of delay elements to an optic fiber on the same azimuth, but located at a larger distance from the center (Fig. 1.30). As the various optic fiber pulses pass through the periphery, the former purely spatial relationships of the figure contour are converted into temporo-spatial relationships. As in Harmon's expanding circular scanning system, the figure is transformed into a space in which size variations appear as time delays, and the effects of rotation are trivial. In fact, by properly storing the delay-space outputs, a binary matrix representation of the input pattern can be obtained which is identical with the binary matrices obtained by Harmon.

Singer has suggested a self-organizing recognition scheme for use with the above input system. During a learning phase, a large number of image code matrices are stored in a ferrite memory plane. Before each new image code is stored, it is tested to make sure that it is sufficiently different from the image codes already stored. In this way, excessive redundancy in the memory is avoided. It should be observed that two forms of the same character (such as "A" and "ᴙ") would be stored as separate images in this system. A control unit later re-examines the memory for such redundancies in order to reduce them by logical analysis.

Recognition proceeds as follows: The complement of the code matrix

FIGURE 1.28
Basic Contour-sensing Unit

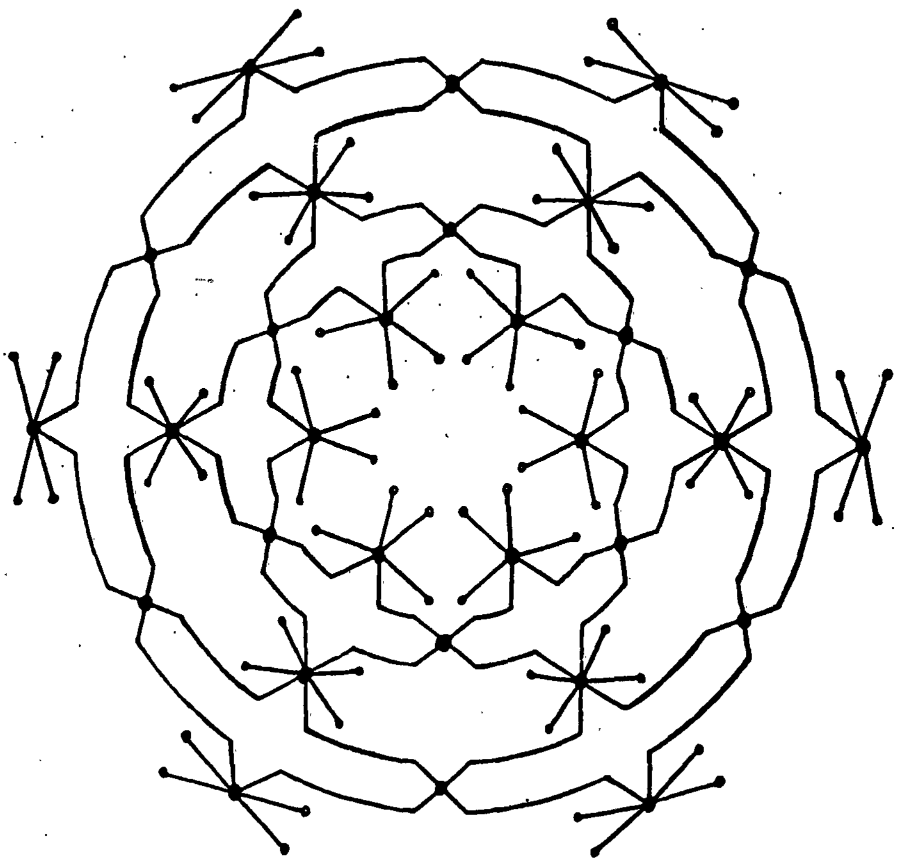




indicates inhibitor input

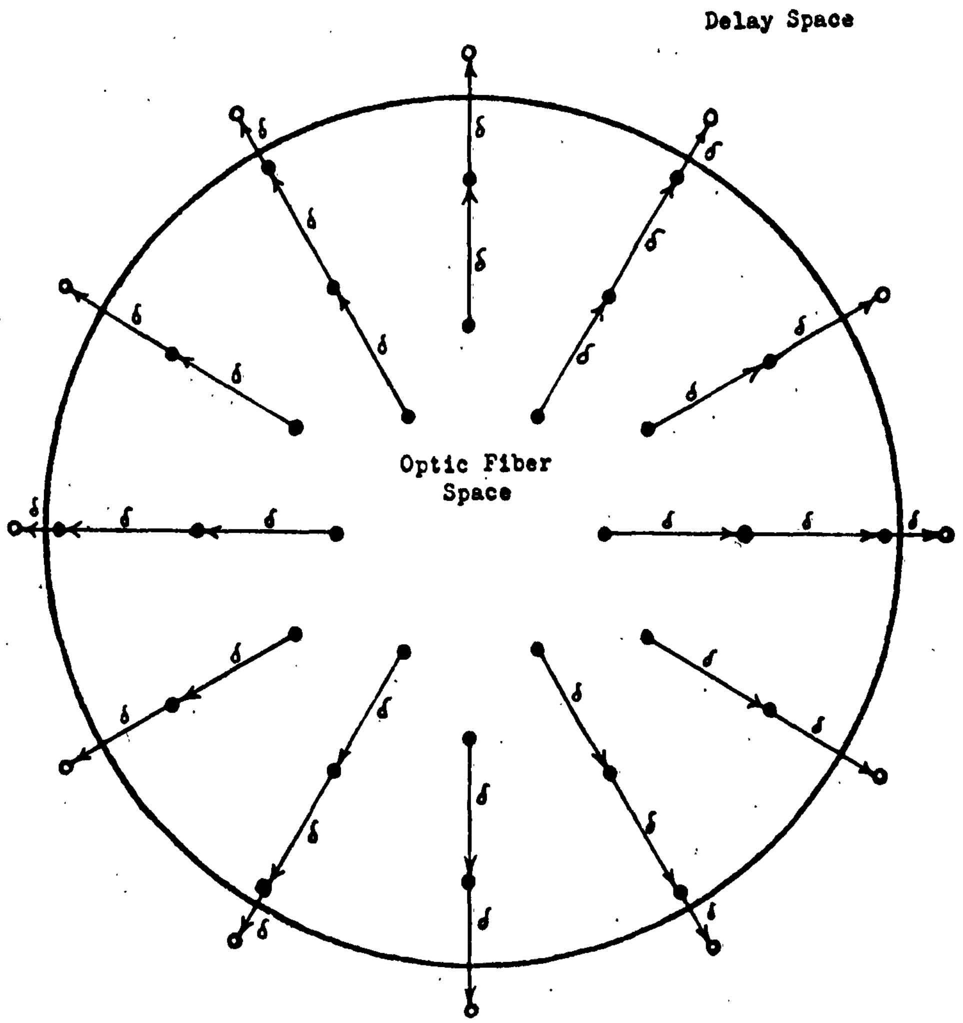
indicates n-input AND gate
(n inputs must be active for output pulse to occur)

FIGURE 1.29



Solid black circles represent optic fibers.
Small white circles represent photosensitive elements.

FIGURE 1.30



δ = delay time
Solid black circles represent optic fibers.
Open circles represent delay space output fibers.

of the image to be recognized is taken. This complement is then added in turn to each of the stored image matrices having the same order as the complement matrix. If the total sum of all the digits in the sum matrix is not within the range $mn \pm e$, where m is the number of rows, n is the number of columns, and e is a "noise number" which is optimized by the machine, then the symbol is not "recognized". In case the input matrix is not recognized as any of the stored matrices, it is stored as a new prototype matrix.

The optimal value of e depends both on the stored images and on the nature of the images to be recognized. If e is too large, misrecognition or multiple recognition will be common; if e is too small, recognition will not be achieved, and the memory will be loaded down with redundant images. A program for optimizing the value of e on the basis of experience gained by processing a large number of samples is incorporated in the machine.

1.5.4 Work Done by Paul Baran, Mel Breuer, and Richard Manelis

This report is a continuation of research work done by Baran, Breuer, and Manelis at the University of California at Los Angeles Digital Technology Laboratory (1,3,18).

Since computer routines and techniques developed by these workers have been used in this report, a brief review and explanation of their work is essential and is provided in this section.

Baran's Work:

The purpose of Baran's thesis was the development and demonstration of a pattern recognizing technique which would be suitable for use with language translation machines. Baran's technique was developed with the following observations in mind:

(1) To be useful, the technique would have to be capable of recognizing punctuation, as well as alphabetic and numerical characters of varying styles and fonts.

(2) A reasonable amount of noise in the form of dcformed characters, misalignments, smears, etc., must be allowable without undue deterioration of reading ability.

(3) The technique would have to involve a learning process in which the reading machine "adapts" itself to the range of type fonts and printing distortions characteristic of the material to be translated. Otherwise the reading machine could only be used to read one particular type

font, whereas often one encounters several different type fonts (e.g. roman, italic, bold-face) on one page of a journal or document.

(4) Accuracy need not be maintained at the level necessary in, for example, a check-reading machine where a single misread digit might be disastrous. The high redundancy of written languages would make error detection simple, particularly if some indication of the likelihood of a reading error were provided by the reading machine.

The theoretical basis of Baran's recognition scheme is as follows: Let us assume that we are dealing with characters which have been expressed in the form of binary matrices by superimposing them on a rectangular grid containing IxJ cells C_{ij} (Fig. 1.13). Let us further assume that we desire to recognize a set of K characters $\{N_1, N_2, \cdots, N_K\}$, whose a priori probabilities of occurrence $\{P(N_1), \cdots, P(N_k)\}$ are known. The recognition scheme consists of two phases. In the preliminary or "learning" phase, we feed t representative samples of each of the K characters into the reading machine (a digital computer). The machine computes the conditional probabilities

$$\left\{ \begin{array}{l} P(b_{ij} \mid N_k) = \sum_{\ell=1}^{t} b_{ij\ell}/t \\ P(w_{ij} \mid N_k) = 1 - P(b_{ij} \mid N_k) = 1 - \left(\sum_{\ell=1}^{t} b_{ij\ell}\right)/t \end{array} \right\} \quad \begin{array}{c} (1 \le i \le I;\ 1 \le j \le J; \\ 1 \le k \le K) \end{array}$$

Where
w_{ij} = the event "cell c_{ij} is white"
b_{ij} = the event "cell c_{ij} is black"

$$k_{ij\ell} = \begin{cases} +1 \text{ if cell } C_{ij} \text{ of the } \ell^{th} \text{ sample of } N_k \text{ is black.} \\ 0 \text{ if cell } C_{ij} \text{ of the } \ell^{th} \text{ sample of } N_k \text{ is white.} \end{cases}$$

The learning phase is completed by inputting the known a priori probabilities $p(N_1), p(N_2), \ldots, p(N_K)$.

In the recognition phase, an unknown character N is fed into the machine. Recognition is accomplished by computing the a posteriori probabilities, (based upon the information contained in the cells of N) of the K hypotheses:

$$H_1 : N_u = N_1$$
$$H_2 : N_u = N_2$$
$$\cdots$$
$$H_K : N_u = N_K$$

The process is completed by assuming that the hypothesis H_k with the highest a posteriori probability is the correct one. The mathematical basis of the digital computation is now given.

Suppose that the first cell c_{11} of N_u is black. On the basis of this cell only, we can use fundamental rules of probability theory to compute the a posteriori probabilities of the hypotheses $H_1, H_2, \ldots, H_K$. We have:

$$P_1(H_k) = P(N_k | b_{11}) = \frac{P(N_k, b_{11})}{P(b_{11})} = \frac{P(b_{11}|N_k)P(N_k)}{\sum_{k=1}^{N} P(N_k, b_{11})} = \frac{P(b_{11}|N_k)P(N_k)}{\sum_{k=1}^{N} P(N_k)P(b_{11}|N_k)} \quad (k = 1, 2, \cdots, K)$$

Similarly, if cell c_{11} of N_u is white, we have

$$P_1(H_k) = P(N_k | w_{11}) = \frac{P(w_{11}|N_k)P(N_k)}{\sum_{k=1}^{K} P(N_k)P(w_{11}|N_k)} \quad (k = 1, 2, \ldots, K)$$

To simplify matters, let

$$X_{b_{ij}} = \frac{1}{P(b_{ij})} = \frac{1}{\sum_{k=1}^{K} P(N_k)P(b_{ij}|N_k)}$$

$$X_{w_{ij}} = \frac{1}{P(w_{ij})} = \frac{1}{\sum_{k=1}^{K} P(N_k)P(w_{ij}|N_k)}$$

The quantities $X_{b_{ij}}$ and $X_{w_{ij}}$ are called respectively the "white cell weight factor" and the "black cell weight factor" of Cij , since they are related to the "weight" or "importance" of the cell Cij. More will be said about the significance of the cell weight factor later.

It is also convenient to introduce a variable Vij such that

$$V_{ij} = \begin{cases} b_{ij} \text{ if cell } c_{ij} \text{ of } N_u \text{ is black} \\ w_{ij} \text{ if cell } c_{ij} \text{ of } N_u \text{ is white} \end{cases}$$

We can then write the single formula

$$P_1(H_k) = P(N_k|v_{11}) = X_{v_{11}} P(v_{11}|N_k)P(N_k)$$

If we make the simplifying approximation that the cells are statistically independent, then we can compute the second-order <u>a posteriori</u> probability $P_2(N_k)$ which is based upon the information contained in the first <u>two</u> cells, by a similar analysis. However, we now use the first-order <u>a posteriori</u> probability $P_1(N_k)$ instead of the <u>a priori</u> probability $P(N_k)$. Hence,

$$P_2(N_k) = P(N_k|v_{11}, v_{12}) = X_{v_{12}} P(v_{12}|N_k)P_1(N_k) = X_{v_{11}} X_{v_{12}} P(v_{12}|N_k)P(v_{11}|N_k)P(N_k).$$

Continuing in this manner, it is evident that the final <u>a posteriori</u> probability of H based upon the entire N_u matrix is given by

$$P_f(N_k) = P(N_k) \prod_{i=1}^{I} \prod_{j=1}^{J} X_{v_{ij}} P(v_{ij}|N_k) . \qquad (I-1)$$

As stated previously, the prime choice for the identity of N_u is the character N_e having the maximum value of P_f . It will be observed that in addition to an educated guess N_e as to the identity of N_u , the recognition scheme yields an estimate of the confidence of the guess in the form of $P_f(N_k)$. Furthermore, by examining the values of $P_f(N_k)$ for $K \neq e$, we can find the most likely alternatives to N_e in case there is reason to suspect this estimate of being incorrect. These properties are very useful in language translation reading schemes, as was noted earlier.

As indicated in the derivation, the Baran scheme is based upon the obviously inaccurate assumption of statistical independence of the cells. However, for reasonably undistorted unknowns, this is not likely to cause errors in recognition. To demonstrate this, Baran programmed the IBM 709 computer to perform the computations indicated above. As input characters, he utilized 96 samples of each of the ten numerals. These samples were in the form of binary digit matrices which had been punched on cards by IBM for computing purposes. Varying degrees of distortion were represented, ranging from near-perfect samples to severely under-inked ones. A few samples of these numerals are shown in Fig. 1.31. Using half of the samples in the learning phase, and the other half as unknowns, Baran was able to achieve a correct recognition rate of 91%. This is certainly sufficient for the purposes of a language translation reader.

Baran considered the possibilities of producing special purpose read-

A few samples of characters used by Baran and Breuer

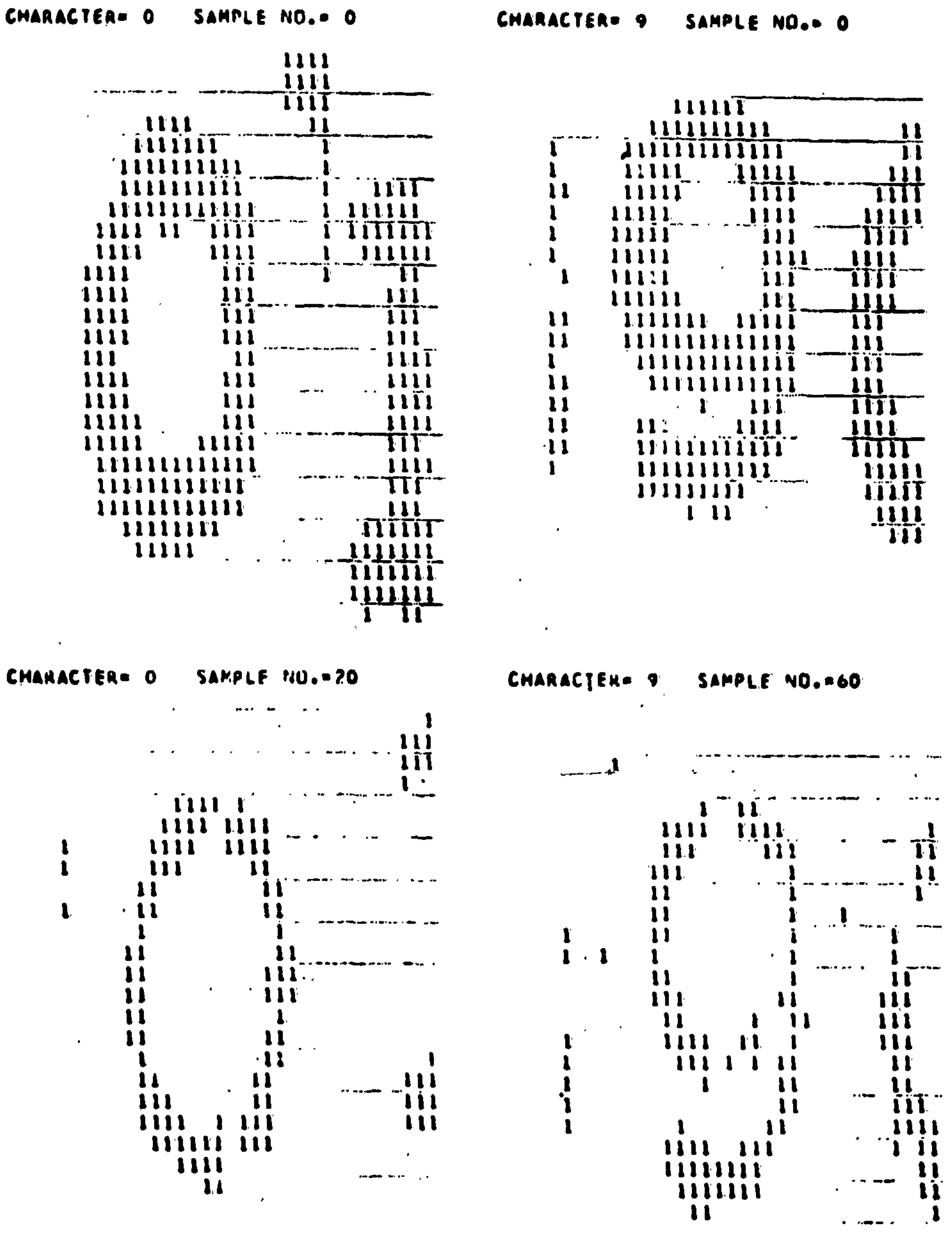

CHARACTER= 0 SAMPLE NO.= 0
CHARACTER= 9 SAMPLE NO.= 0
CHARACTER= 0 SAMPLE NO.=20
CHARACTER= 9 SAMPLE NO.=60

FIGURE 1.31

ing machines using optical filters based upon his recognition scheme. Readers interested in this, or in the details of the computer program, the character samples, etc., are referred to his thesis.

Breuer's Work:

Breuer's thesis covered a broad range of topics, including an alternate (masking) recognition scheme, computer routines to aid recognition by filtering and centering input characters, and a "binary division" routine for finding the most significant areas of the character field from the standpoint of differentiating the characters. This binary division routine, in addition to other work done by Breuer on the significant area problem, is discussed elsewhere in this chapter.

Breuer's recognition scheme is actually a simplified form of Baran's scheme. As in Baran's scheme, there is a preliminary learning phase in which representative samples of each of the characters to be recognized, expressed in the form of binary matrices, and properly identified, are fed into the machine. The conditional probabilities $P(b_{ij}|N_k), P(w_{ij}|N_k)$ are estimated by the machine as discussed previously. Recognition of an unknown character N_u is based upon the simple formula

$$P(H_k) = P(N_k = N_u) = \prod_{i=1}^{I} \prod_{j=1}^{J} p(v_{ij}|N_k)$$

Where $v_{ij} = \begin{cases} b_{ij} \text{ if cell } c_{ij} \text{ of } N_u \text{ is black} \\ w_{ij} \text{ if cell } c_{ij} \text{ of } N_u \text{ is white} \end{cases}$ as defined previously.

The unknown character N_u is identified as N_e such that

$$P(H_e) > P(H_k) \quad for\ k \neq e.$$

It is evident that the above formula is actually an expression for $P(\tilde{V}|N_k)$, where $\tilde{V} = (v_{11}, v_{12}, \cdots, v_{1J}, v_{21}, v_{22}, \cdots, v_{IJ})$ is a vector representing the observed state of the cells of N_u. (Again the approximation of independence of the cells is made). In general, one would not expect Breuer's scheme to be as accurate as Baran's scheme, which yields values for the more pertinent quantities $P(N_k|\tilde{V})$. Inasmuch as Breuer's scheme makes no provision for variations in the a priori probabilities of the individual patterns, it may break down if the patterns are not equi-probable. However, as Breuer showed experimentally, his recognition scheme is about as accurate as Baran's for equi-probable patterns in which the distortion rates are not excessive.

Breuer programmed his recognition scheme in the form of a masking technique by taking the logarithms of the quantities $P(\ell_{ij}|N_k)$ and $P(w_{ij}|N_k)$. Recognition is based on the formula $\log P(H_k) \cong \sum_{i=1}^{I} \sum_{j=1}^{J} \log P(v_{ij}|N_k)$.

Since the logarithm is a monotonically increasing function, recognition can be based on the value of log $P(H_k)$ just as well as on that of $P(H_k)$. Provision must be made for the cases $P(w_{ij}|N_k) = 0$ and $P(\ell_{ij}|N_k) = 0$, since the logarithm of 0 is not defined. This was done by adding a small positive amount of "noise" η to all of the a posteriori probabilities $P(\ell_{ij}|N_k)$ and $P(w_{ij}|N_k)$ before taking logarithms.

Breuer simulated several filtering schemes designed to improve recognition by removing random ink blotches, filling in missing portions of the character, and smoothing the edges. He also programmed a centroidal translation routine which shifts an input character so that its centroid coincides with the center of the scanning field. The usefulness of these routines was established by Breuer in an experiment in which a reduction in error rate of about 30% was achieved through their use.

Manelis' Work:

The object of Manelis' thesis was to develop computer routines for the generation of character samples having controlled or random amounts of distortion, starting with a set of ideal characters. The characters so generated are useful inputs for testing character recognition schemes. In fact, the impetus for development of the character generation routines arose from the inadequacy of the sample numerals used by Baran and Breuer for conclusive testing of their character recognition techniques.

Specifically, the samples used by Baran and Breuer had the following shortcomings: First, the number of samples used was necessarily limited to the number available on punched cards (96 samples of each numeral). As these samples alone required 28,800 cards, it is obvious that the number of samples was not readily extendable. For statistical reasons, however, it may be desirable to have a much larger number of samples available for both the learning and recognition phases when testing recognition rates. Secondly, the available samples represented only one of many possible types of deviation from the ideal, viz. that due to a more or less worn out typewriter ribbon. Hence, if one desired to test a recognition system designed, say, to operate in a milieu of over-inked rather than under-inked characters, the numeral samples of Baran and Breuer would obviously not be ideal guinea pigs. Finally, the only means Baran and Breuer had of controlling the amount of distortion prevalent in their samples was through increasing or decreasing the number of samples used. Increasing the number of samples increased the average amount of distortion, and vice versa, since the samples became progressively more deteriorated as the sample numbers increased. However, this provided only a rough, qualitative control over the amount of distortion present. Some sort of quantitative measure of the distortion rate, on the other hand,would be very useful as it would permit the experimenter to make objective statements about the ability of his recognition scheme to read distorted characters successfully. By measuring the amount of distortion present in a typical document, the experimenter would be able to predict whether or not his reading machine was accurate enough for the intended purpose.*

All of the above shortcomings can be obviated by use of character generation techniques of the type developed by Manelis. Manelis started with a set of 33 ideal Cyrillic (Russian) characters which he quantized and

* In case the reading machine incorporates a filter and/or smoother, the experimenter would, of course, measure the distortion rate of the output of the filter and/or smoother when the input characters are taken from the document in question.

punched on IBM cards in the same manner as the numeral samples used by Baran and Breuer. A set of (10 of) the quantized ideal characters are shown in Fig. 1.32. Manelis developed several types of character distortion routines. Two of these are controlled displacement routines whose purpose is to shift the ideally centered character matrix a specified number of columns to the right or to the left; two others shift the matrix vertically by a fixed amount at the discretion of the experimenter. The results of applying these programs to ideal characters are shown in Fig. 1.33. Manelis also developed a random shifting routine whose purpose is to shift the input character by a random amount horizontally and vertically in succession. The amount of shifting in this routine is determined by a sub-program which generates random numbers. These shifting routines simulate the effects of systematic and random errors which might occur in the character-centering mechanism of a reading machine.

Another routine by Manelis generates "random unbiased distortion". This involves changing the color of a specified percentage of the cells in the character matrix. A random number generating program determines which cells are to be affected. The results of this routine are demonstrated in Fig. 1.34 for eighteen percent distortion. In practice, this type of distortion might arise as a result of such causes as paper roughness, electrical noise in a photocell scanning mosaic, or typing through carbon paper.

Manelis' Ideal Cyrillic Characters

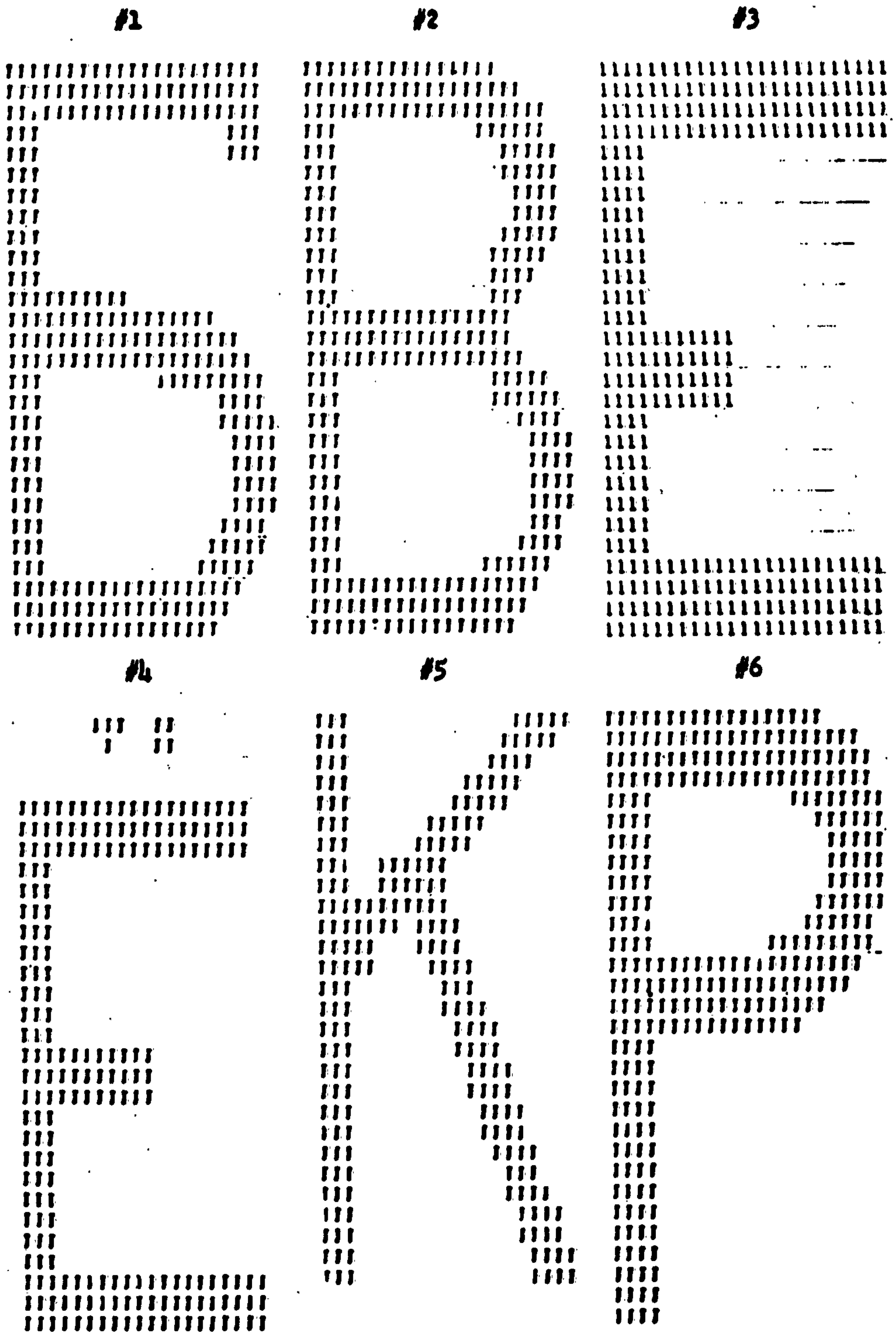


FIGURE 1.32

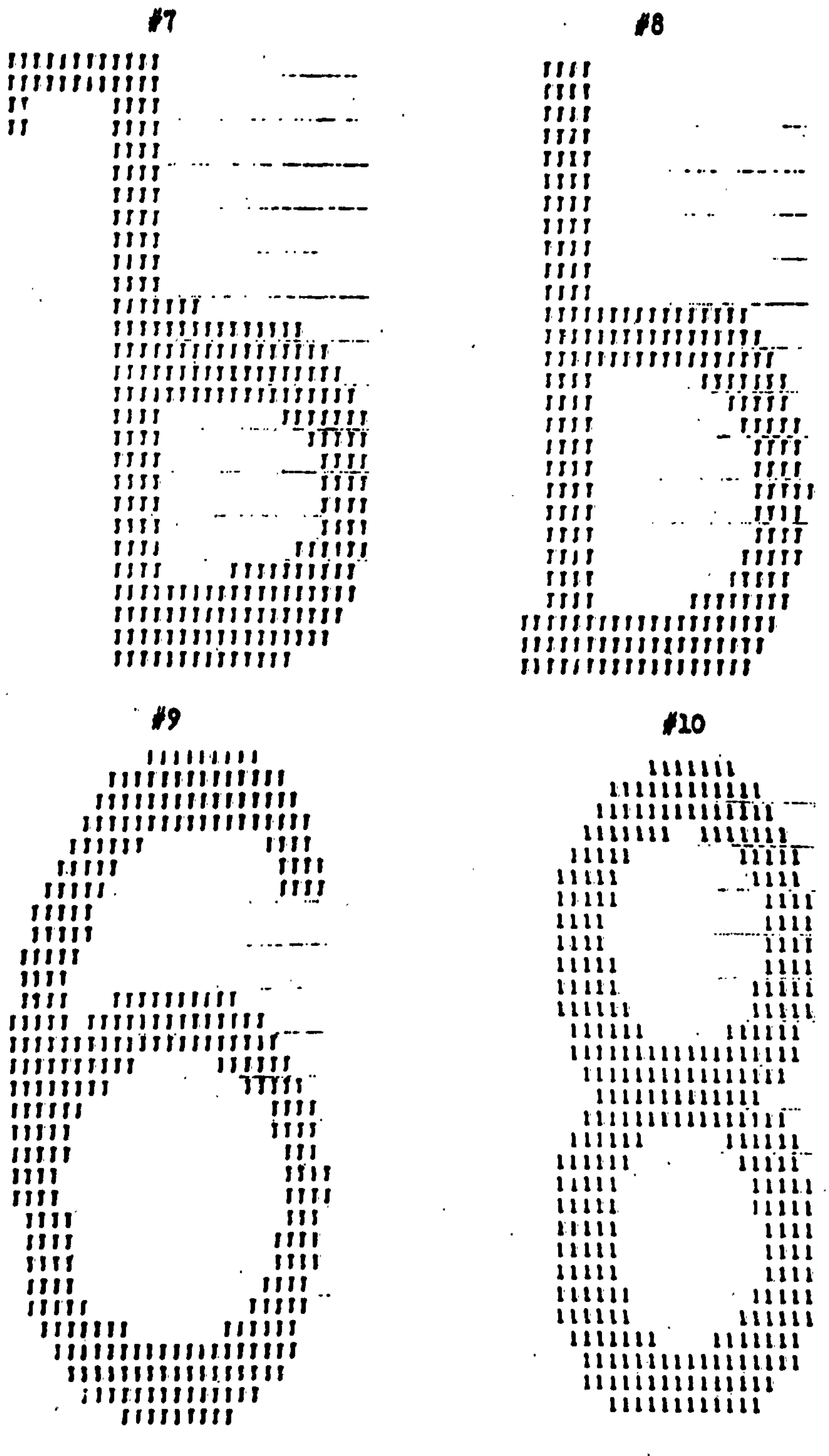


FIGURE 1.32 (CONT.)

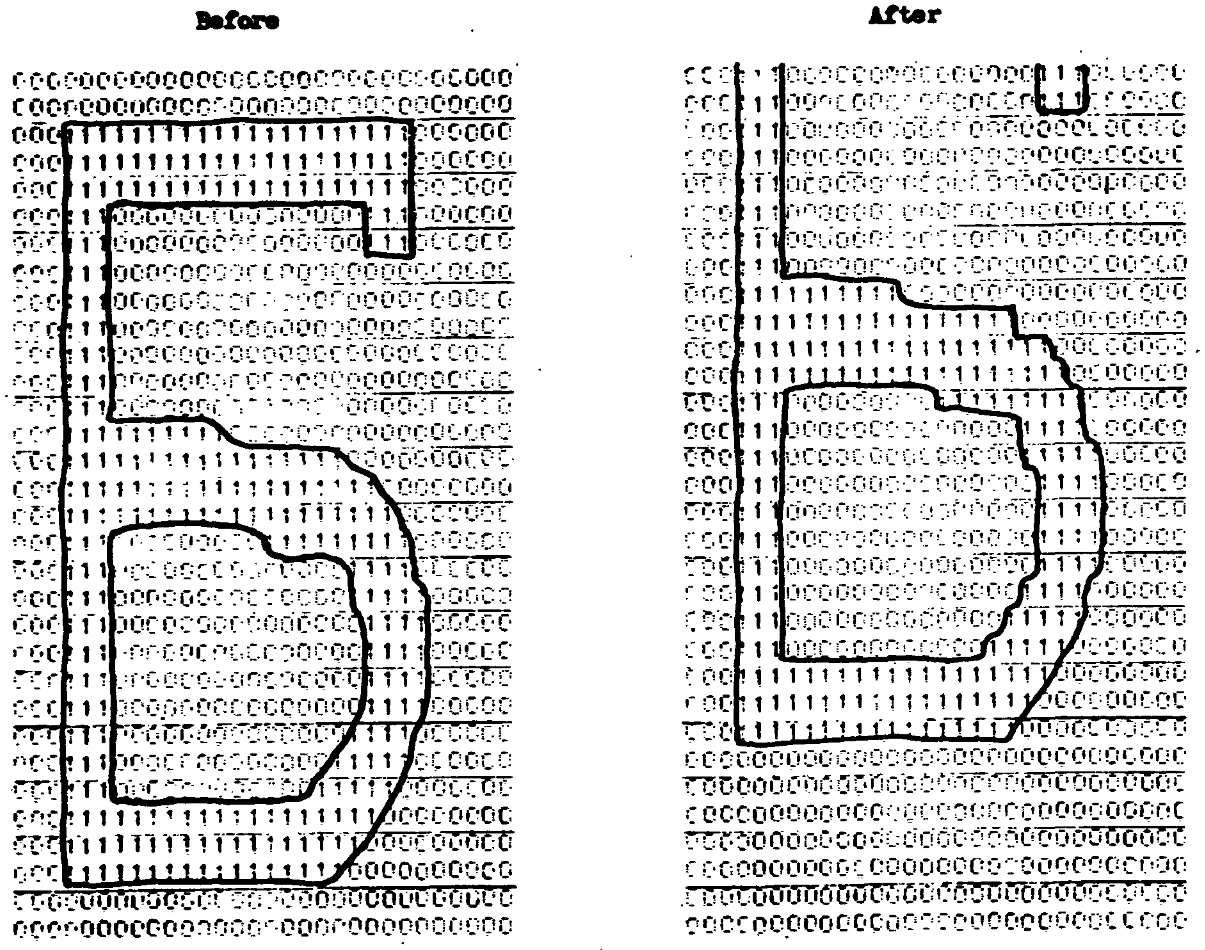


FIGURE 1.33

Manelis' vertical shift routine 5 rows upwards

Before

After

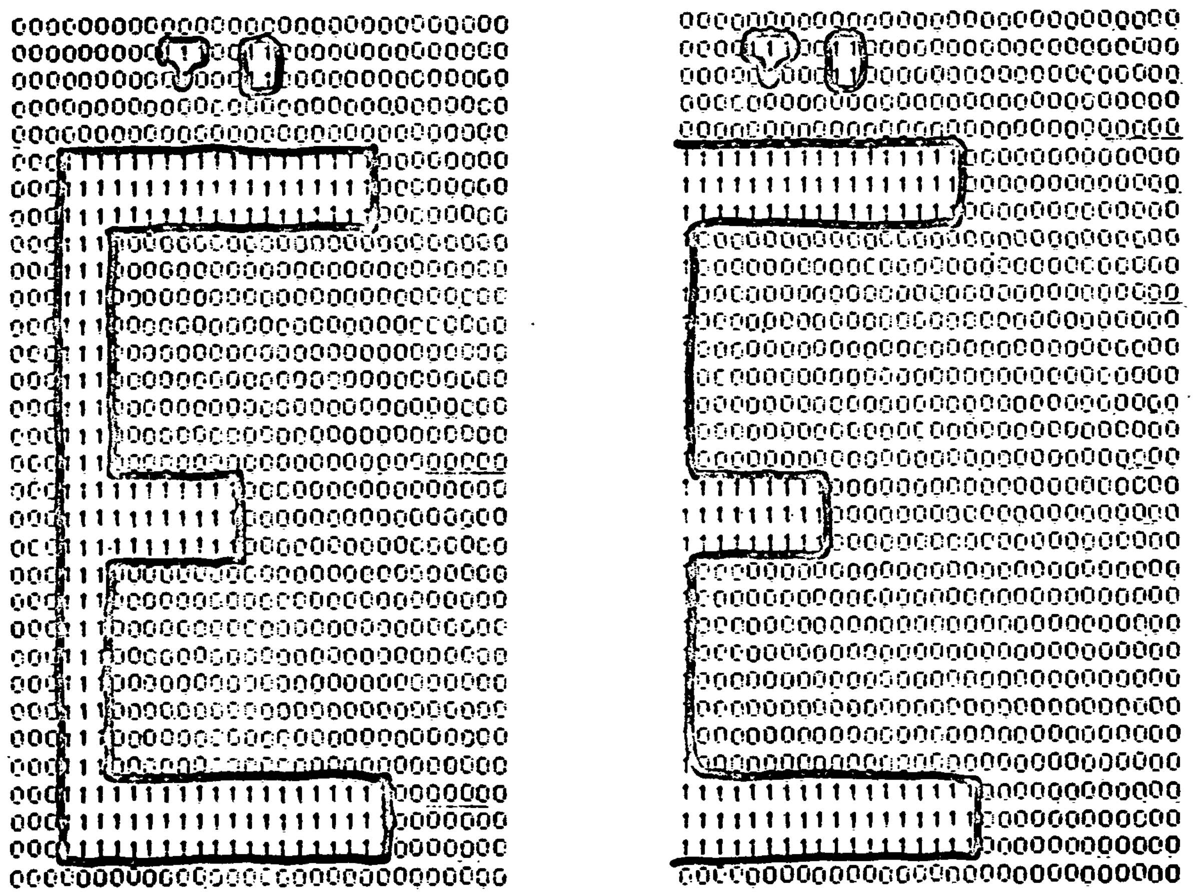


FIGURE 1.33 (CONT.)

Manelis' horizontal shift routine 5 columns to the left

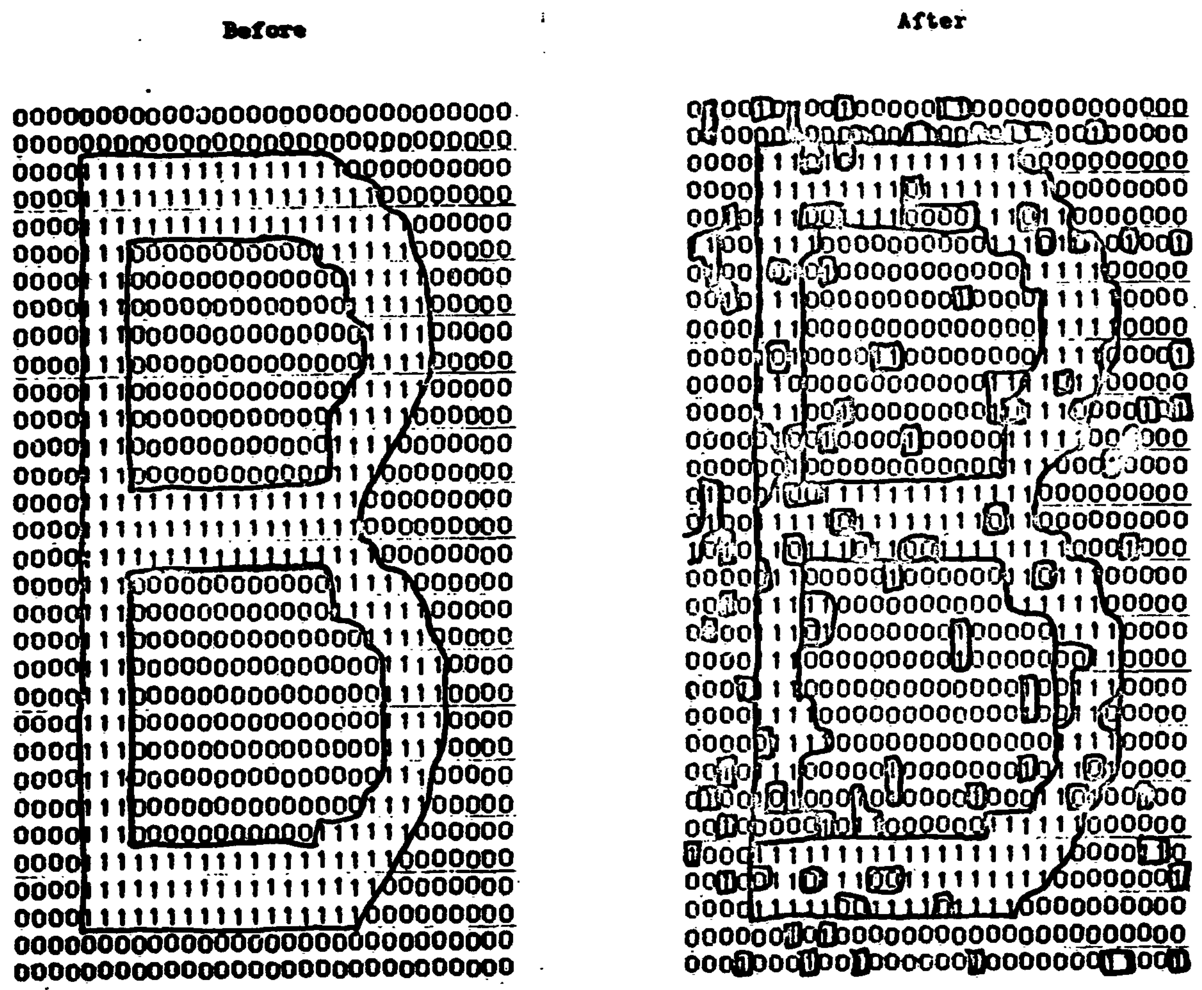


FIGURE 1.34

18% random unbiased distortion

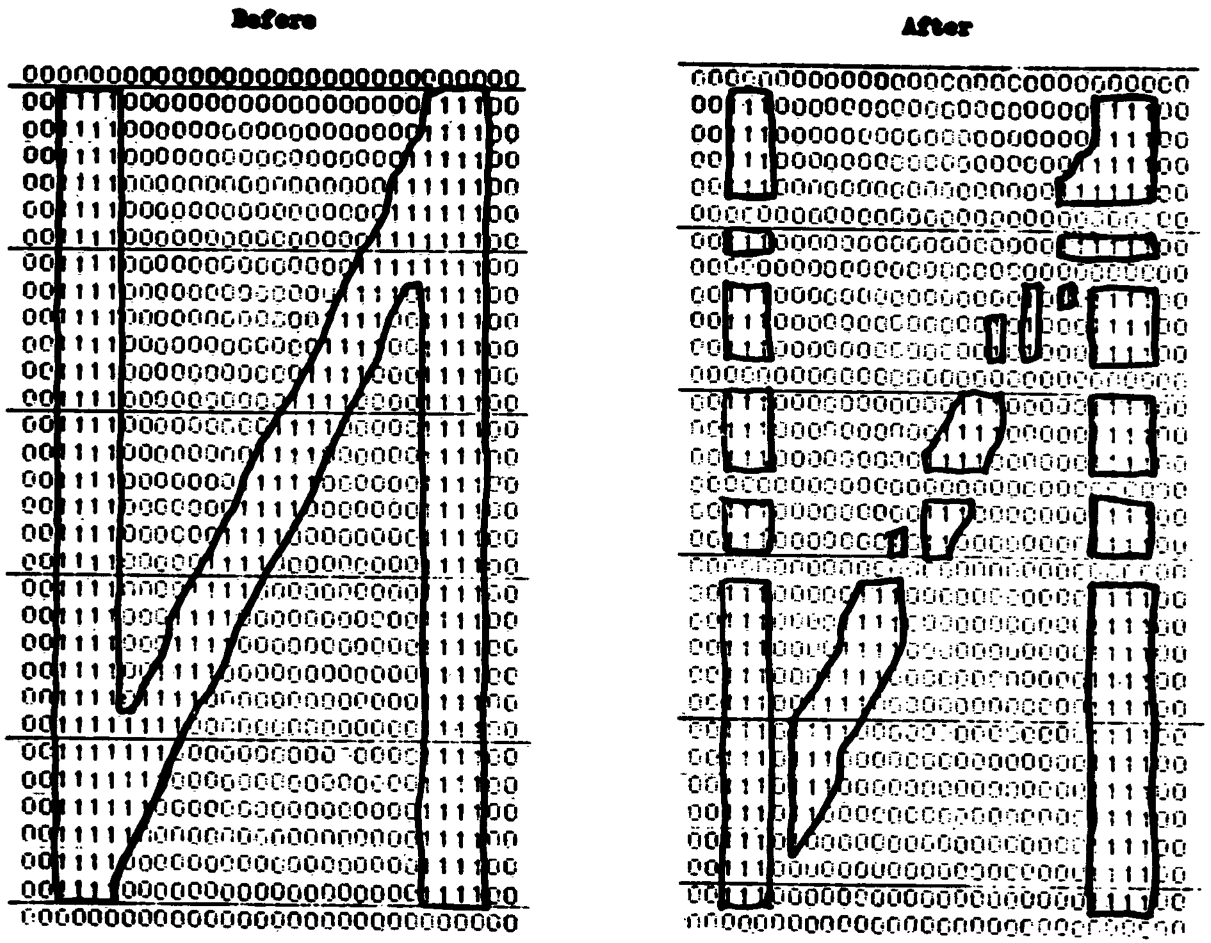


FIGURE 1.35

20% random biased distortion

A final type of distortion simulated by Manelis is known as random biased distortion (Fig. 1.35). A specified number of entire rows and/or columns are deleted (i.e., set equal to zero). A random number first determines whether a row or a column is to be deleted. A second random number determines which row (or column) is to be deleted. Presumably, distortions of this type arise in typewriting when the ribbon has been heavily used.

By combining the above types of distortion and by varying the degrees of the individual distortions, many of the noise situations apt to be encountered in practice can be adequately simulated.

1.5.5 Review of Recent Literature Pertaining to Property Significance

In this section, work done by previous authors relating to the problem of finding significant pattern-discriminating properties will be reviewed. In the literature, the problem has so far been approached from the standpoint of finding minimal relative code representations for the unknown characters. In other words, given a set of prototype characters, the question is asked: "Taking into account possible variations due to noise, by what means can the characters be coded so that (1) the codes for different characters will almost always be distinct, and (2) the codes will be as short as possible?" As mentioned previously, hardware and recognition time can often be reduced substantially by solving this problem.

The minimal representation problem is closely related to the problem of determining the significance of properties or areas of the pattern.

This can be seen as follows: each digit in the code representation of a character indicates the presence of a specific property in the character. It follows, therefore, that the less significant the properties corresponding to the code digits are, the less information each code digit will contain concerning the identity of the character represented, and the greater the code length will have to be in order for the characters to be uniquely represented. Moreover, given a non-minimal coding scheme for a group of patterns, the problem of minimizing the code by eliminating redundant or irrelevant digits is equivalent to the problem of reducing a set of properties by finding and eliminating the redundant and irrelevent (i.e., "non-significant") ones.

It has been pointed out previously that no generality is lost by restricting the discussion to the determination of significant <u>areas</u> of the patterns, rather than significant <u>properties</u>. For simplicity, we shall make this restriction in the remainder of this report.

Arthur Glovazky (11) attacked the problem of determining minimal code representations for noise-free patterns satisfying the conditions:

(1) The number of patterns P is finite.

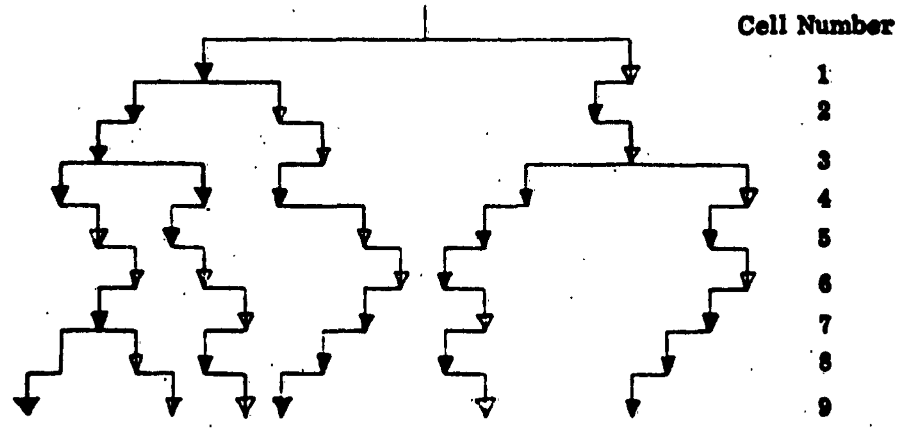
(2) All of the patterns are of the "two-tone" (black-and-white) variety, and

(3) any of the given patterns can be divided into a finite number of cells C, each of which is either wholly black or wholly white. Glovazky restricted his discussion to C-digit binary codes formed by numbering the cells in an arbitrary sequence called (for obvious reasons) the

"scanning path", and then setting the i^th digit of the code equal to $\left\{ \begin{matrix} \text{zero} \\ \text{one} \end{matrix} \right\}$ = if the i^th cell is $\left\{ \begin{matrix} \text{white} \\ \text{black} \end{matrix} \right\}$. Glovazky gave two methods, each of which enables one to minimize a given scanning path by eliminating superfluous cells. The reduced number of cells obtained is the minimum permissible number for the given scanning path. It is quite possible that a different scanning path will yield a smaller number of cells, and therefore a more economical identification process.

Glovazky's first procedure involves the construction of a code mobile, a figure quite similar to the "character trees" discussed previously.

FIGURE 1.36
The "Code Mobile"



A typical code mobile, for the case P=6, C=9, is shown in Fig. 1.36. Each of the nine "levels" of the mobile corresponds to inspection of a specific cell of the pattern. Each "chain" of the mobile represents the results of scanning one of the six patterns according to the assumed scanning path. At each level, the chain is diverted to the $\left\{\begin{matrix}\text{right}\\\text{left}\end{matrix}\right\}$ if the pattern is $\left\{\begin{matrix}\text{white}\\\text{black}\end{matrix}\right\}$ in the corresponding cell. The code mobile shown therefore corresponds to patterns with binary codes as follows:

	1	2	3	4	5	6	7	8	9
1	1	1	1	0	1	0	0	1	0
2	0	1	1	0	1	0	1	1	1
3	0	1	0	1	1	1	0	1	0
4	1	0	0	1	0	0	1	1	1
5	1	1	1	1	0	0	1	0	0
6	1	1	1	1	0	0	1	1	1

Table 1-1

The points of interconnection of the chains are called "nodes". The presence of a node at a given level of the mobile indicates that the corresponding cell in the scanning path is relevant and cannot be eliminated. If, on the other hand, there are no nodes at a given level, then the corresponding cell can be eliminated from the scanning path without impeding the recognition process in any way. Inspection of Fig. 1.36 reveals that cells 2, 3, 5, 6, 7 and 9 can be eliminated. The resulting reduced code mobile is shown in Fig. 1.37, and the reduced binary codes for the patterns are:

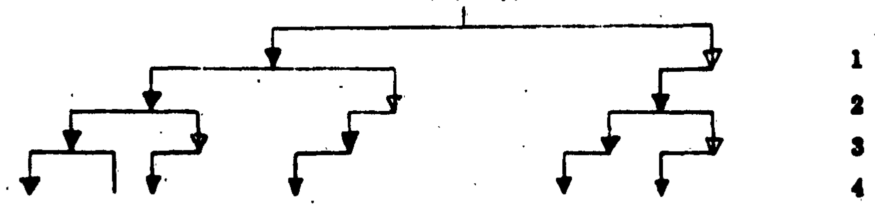
	1	2	4	8
6	1	1	1	1
5	1	1	1	0
1	1	1	0	1
4	1	0	1	1
3	0	1	1	1
2	0	1	0	1

Table 1-2

The validity of the above reduction can be confirmed by observing that no two rows of Table 1-2 are identical, and that no column can be deleted without destroying the distinctness of the rows.

A necessary and sufficient condition for recognition of the patterns in the minimum possible number of cells $\{\log_2 P\}$, is clearly the following: at each level of the code mobile, each chain must branch. It follows immediately that at the n^{th} level, there must be 2^n nodes. It is also evident that if $P=2^\gamma$, where γ is an integer, each cell in the absolute minimal scanning path must be black for <u>precisely</u> one half of the characters and white for the remaining half. This statement is approximately true when $P \neq 2^\gamma$.

FIGURE 1.37



Reduced Code Mobile

In practical problems, construction of the code mobile may prove very cumbersome due to large values of P and C. Glovazky's second method, which is actually equivalent to the code mobile procedure, is more useful under such circumstances. The binary codes are first arranged in a sequence of descending numerical value, known as a "code schedule" (Table 1-3).

$$\begin{array}{c|ccccccccc}
 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\
\hline
6 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & \underline{1} & \underline{1} \\
5 & 1 & 1 & 1 & \underline{1} & \underline{0} & \underline{0} & \underline{1} & \underline{0} & \underline{0} \\
1 & 1 & \underline{1} & \underline{1} & \underline{0} & \underline{1} & \underline{0} & \underline{0} & \underline{1} & \underline{0} \\
4 & \underline{1} & \underline{0} & \underline{0} & \underline{1} & \underline{0} & \underline{0} & \underline{1} & \underline{1} & \underline{1} \\
3 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 0 \\
2 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 1
\end{array}$$

Table 1-3

Code Schedule for Table 1-1 and associated separation lines.

The procedure consists in finding the points on the code schedule corresponding to the nodes of the code mobile. Proceeding from left to right, the first column is found in which 1 neighbors 0 (column 1 in Table 1-3). A <u>separation line</u> is then drawn between the two digits extending from the column in question to the right as shown. The arrays on each side of this line can now be regarded as new schedules. The process is repeated (Table 1-3) until all the columns are exhausted or each of the resulting "subschedules" contains only one row.

The points at which the separation lines begin correspond to the nodes of the code mobile. Hence, all columns which do not contain such points can be eliminated, yielding again the reduced code schedule of Table 1-2.

It so happens that the patterns represented by Table 1-2 can be identified by scanning only three cells. It can be readily verified that either cells 3, 4 and 9 or 3, 5 and 9 (in each case one cell less than the minimum of 4 cells necessary for the scanning path considered above) are sufficient to identify the pattern. The problem of finding a scanning path which can be reduced to an absolute minimal scanning path is very difficult, at least for practical magnitudes of P and C. Using Glovazky's method, one would have to examine all possible cell sequences, a process far too lengthy to be feasible. However, it can be shown that the reduced scanning sequences obtained by Glovazky's method are at most P-1 cells in length. Therefore, for cases in which $C \ll (P-1)$, and P-1 is a sufficiently small number, Glovazky's method always yields satisfactorily reduced cell sequences. Moreover, one can greatly increase the chances of finding an absolute minimal scanning path by placing at the beginning of the code schedule those columns in which the ratio between "ones" and "zeros" is closest to unity (i.e., those columns with the highest degree of uncertainty.) (10). This method breaks down, however, whenever this ratio is close to 1 for all columns. Under these circumstances, unless the scanning path is already minimal, one finds that there is a great deal of duplication of information by the various cells. In other words, there is a large amount of redundancy in the system. Additional rules for improving the effectiveness of the code schedule technique have been developed by O. Lowenschuss (17) and by M. Paull (20).

Arthur Gill (19) gives a rather lengthy method requiring computer processing, which always yields the absolute minimal scanning path. Briefly, the process is as follows: one begins by considering an arbitrary group of P patterns composed of C cells and the associated PxC "code schedule" of the type shown in Table 1-1. Assuming first that only the pattern represented by the first row of the code schedule is to be recognized, it is evident that any of the cells 1 to C may be selected as a legitimate path. Suppose now that pattern 2 is added to the set. Then each of the paths selected previously may or may not be still adequate, depending on whether or not the part of pattern 1 in that path coincides with the part of pattern 2 in that path. If such a coincidence occurs, the path in question has to be augmented by adding to it another cell in which patterns 1 and 2 differ. Carrying out all possible augmentations in this manner, a revised list of paths is obtained, including all paths adequate for recognition of patterns 1 and 2. Next, pattern 3 is added to the set, and the augmentation process is repeated, yielding a list of paths adequate for recognition of the first <u>three</u> patterns. The process is continued until the last row of the code schedule has been added to the set, at which point a list of all possible paths (consisting of at most P cells) which are adequate for recognition of the P patterns is obtained. The scanning path(s) containing the least number of cells can be easily found by inspection of the list.

Gill has formulated the above process in a manner suitable for computer programming. In addition, he has developed a more general procedure to deal with the "noisy" case in which either the patterns or the scanner is imperfect. In the noisy case, it is desirable to maintain a minimum "distance" $d > 1$ between the codes representing the various patterns. In other words, a scanning path must be found such that any two patterns will differ in at least d of the cells in the path. If this condition is satisfied, it is always possible to detect d-1 errors or correct $\{d/2\}-1$ errors in a given pattern. (12)

The procedure for finding a minimal scanning path consistent with a prescribed d--if such a path exists--is rather involved and will not be given here.

It will be recalled that a necessary (but <u>not</u> sufficient) condition for the recognition of 2^{γ} patterns with the absolute minimal number γ of cells is that the cells employed have the "binary division" property, i.e., the cells must be black for half of the characters and white for the remainder. This fact was used by Breuer (3) as the basis of his search for significant cells. Inasmuch as Breuer dealt with a set of highly distorted characters

(Fig. 1.31), it was necessary to employ an integer threshold to determine whether a cell should be regarded as black for a given character, white for the character, or neither black nor white. More precisely, Breuer defined a cell c_{ij} to be "almost always" black for the character N_k if cell c_{ij} was black more than $(t-\tau)$ times out of a set of t representative samples of N_k. If c_{ij} is black in less than τ of the samples, it is declared "almost always white". If neither of these conditions holds, the cell is said to be "neutral" or "grey" in N_k. Of course, the inequality $\tau < t/2$ must hold.

Assume that K, the total number of prototype characters, is even. Then in Breuer's formulation, a cell is said to have the "binary division" property if it is almost always white for $K/2$ characters and almost always black for the remaining $K/2$ characters. Breuer programmed the IBM 709 computer to determine, for various thresholds, which cells have the binary division property. Breuer's routine also produces a table in which the number of times each cell was black for each character is indicated.

Breuer considered the possibility of constructing a direct-logic recognition system based upon "almost always white" and "almost always black" cells. In such a system, the "character tree" is constructed in the same manner as previously, with the "almost always white" and "almost always black" conditions replacing the "always white" and "always black" conditions used then. At each testing point, therefore, a worst-case probability of error (τ/t) must be reckoned with. The worst-case probability of <u>correct</u> recognition after n such tests is obviously Γ_τ^n, where $\Gamma_\tau = (1 - \tau/\ell)$. It can be seen that the worst-case correct recognition rate in such a system drops rapidly with increasing τ, especially for large values of n.

In order to increase the reliability of the recognition process, Breuer proposed using redundant cells in various ways. For example, if there exist <u>two</u> logical representations for a character N_k, namely $N_k = f_1(C_1, C_2, C_3, C_4)$ and $N_k = f_2(C_5, C_6, C_7, C_8)$, where f_1 and f_2 are arbitrary Boolean functions, then the recognition reliability for this character can be increased by testing all of the cells $C_1, C_2, \ldots, C_8$ and identifying N_k according to the formula $N_k = f_1(C_1, C_2, C_3, C_4) \cup f_2(C_5, C_6, C_7, C_8)$. Breuer showed that this scheme results in a worst-case recognition rate of $\Gamma_\tau^4(2 - \Gamma_\tau^4)$ instead of Γ_τ^4 as in the simple 4-cell schemes. Since $\Gamma_\tau < 1$, this always represents an increase in reliability.

Of course, it is not necessary for a cell to have the binary division property in order to employ it in a recognition scheme. All that is really necessary is that the cell have <u>some</u> discriminating power. That is, the cell must be almost always white for at least one character, almost always black for at least one character, and (possibly) neutral for one or more characters. Such cells were termed "pseudo-significant" by Breuer, and provision was made in the binary division routine for detecting them.

Because of the fact that his character samples were highly distorted, Breuer's binary division routine yielded rather disappointing results. Substantial numbers of binary-divisioncells could only be found when very large thresholds were used. In Fig. 1.38 the binary division cells for $\tau \leq 5$ found by Breuer for 18 samples (1, 3, . . . , 35) of the IBM numerals are shown. The number in each binary division cell indicates the minimum value of τ for which the cell satisfies the binary division criterion. The samples had been filtered to remove noise and aligned with Breuer's centroidal translation program.

Figure 1.39 shows the "pseudo-significant" cells found by Breuer, based upon the same set of samples as Fig. 1.38. The "binary division" cells are, of course, automatically included in this set. The maximum threshold value used was $\tau = 3$. The significant areas are located, as one might expect, away from the border areas of the field, which are white for all characters. The right-hand border, however, is excepted from this rule due to the large amounts of non-random noise occurring in this region (Fig. 1.31).

Breuer suggested that a recognition scheme (e.g., his masking routine) could recognize the characters by considering only the highly significant binary-division or (if necessary), the "pseudo-significant" cells. Presumably, ignoring the other cells would not result in sufficient information loss to impair recognition. Due to time limitations, however, Breuer did not actually test this hypothesis experimentally.

Because of the high threshold values necessary, Breuer did not con-

```
 1
 2
 3
 4
 5
 6                            55
 7            5              5
 8
 9       5
10            54
11
12
13           4        5
14            4
15         5  5
16         34
17                    4
18
19                    4
20       5  5         5
21       5
22            4  5 5
23             5
24                   5
25
26
27           3
28            1233
29
30
31
32
   123456789111111111122222222223
            012345678901234567890
                 I
```

Plot of Breuer's Significant Cells

Maximum threshold = 5

FIGURE 1.38

```
 1
 2
 3
 4
 5
 6
 7             x               xx
 8         xx  x              xxx
 9        xxxxxxxxxxxx         x
10         xxxxxxxxxxxx        xx
11         xxxxxxxxxxxx       xxx
12          xxxxxxx xxxx      xx
13          xxx  x  xxxx       x
14          xxxxxxxxxxxxx    xxxx
15         xxxxxxxxxxxxxx    xxxx
16         xxxxxxxxxxxxxx     xxx
17          xxx  xxxxxxx      xx
18         xxxxxxxxxxxxxx     xx
19         xxxxxxxxxxxxx      xxx
20          xxxxxxxxxx x      xxx
21          xxxxxx xx xx
22           xxx xxxxxx        x
23         xxxxxx xxxxx        x
24          xxxx     xx
25          xxxxxxxxxx
26           xxxxxxxxx
27           xxx
28            xxxx
29
30
31
32
   123456789111111111122222222223
            012345678901234567890
                 I
```

Pseudo-Significant Cells

Maximum threshold factor = .3

FIGURE 1.39

sider it worthwhile to attempt recognition by means of a direct logic system based upon the binary division or "pseudo-significant" cells.

Breuer also investigated the use of Baran's cell weight factors X_{wij} and X_{bij} (see Sec.1.5.4, "Baran's Work") as a measure of cell significance. Unfortunately, Breuer's analysis is very unclear and no conclusions concerning the validity of the cell weight factor were explicity stated. As a matter of fact, the cell weight factors are an extremely poor measure of cell significance. For example, consider the following two distributions of black cells found in t samples of each of 10 characters:

Cell	X_{wij}	X_{bij}	N_1	N_2	N_3	N_4	N_5	N_6	N_7	N_8	N_9	N_{10}
1	2	2	t	t	t	t	t	o	o	o	o	o
2	2	2	t/2	t/2	t/2	t/2	t/2	t/2	t/2	t/2	t/2	t/2

Clearly, cell 2 has absolutely no significance; cell 1, on the other hand, has the binary division property with zero threshold, and hence is of great significance. Nevertheless, the black and white cell weight factors for these cells are identical. This is ample proof of the uselessness of the cell weight factors as a measure of cell significance.

A quite different approach toward the minimal representation problem has been taken by E. L. Blokh (2). Blokh based his work on the following assumptions:

(1) The number of patterns K is finite.

(2) The patterns are undistorted and free of noise.

(3) The patterns consist of C cells, each of which may exist in one of m distinct states or "colors".

(4) Each pattern N_i is assumed to have an a priori probability of occurrence P_i $(i=1,2,\ldots,K)$.

In the minimal representation techniques discussed above, the code length for each pattern was the same. Blokh's method, on the other hand, permits non-uniform code lengths for the various patterns. A scanning method is devised, if possible, such that the expected value of the code lengths is minimum. This amounts to finding a scanning path which permits the most frequent patterns to be identified on the basis of a small number of cells. Larger numbers of cells would have to be scanned in order to identify the less frequent patterns. Of course, to avoid confusion, the code for one pattern must never be identical with the leading digits of the longer code representing a less common pattern.

Blokh's procedure is as follows: The uncertainty in bits associated with each cell c_i is computed according to the formula

$$H(c_i) = -\sum_{j=1}^{m} p(j_i) \log_2 p(j_i) \ ,$$

where $p(j_i)$ denotes the probability that c_i has the j^{th} color. As the first cell k_1 of the scanning sequence, one chooses that cell having the greatest uncertainty value. In case of a tie, an arbitrary choice is made. The second cell k_2 is chosen such that its conditional uncertainty $H(k_2 | k_1)$ given that the color of k_1 is known, is greater than that of all the other cells.

Again, an arbitrary choice is made in case of a tie. The conditional uncertainties are found according to the formula

$$H(c_i | k_1) = -\sum_{l=1}^{m} \sum_{j=1}^{n} p(j_i, l_{k1}) \log_2 p(j_i | l_{k1}),$$

where $p(j_i, l_{k1})$ denotes the joint probability that c_i has the j^{th} color and k_1 has the l^{th} color, and $p(j_i, l_{k1})$ denotes the conditional probability that c_i has the j^{th} color given that k_1 has the l^{th} color. The process is continued, until some number r of cells have been chosen such that $H(k_r | k_1, k_2, \ldots, k_{r-1}) \neq o$ and $H(k_{r+1} | k_1, k_2, \ldots, k_r) = o$ for all choices of k_{r+1}. This condition insures that the r cells are always sufficient to identify the unknown pattern uniquely. The resulting codes for the patterns can be shortened by eliminating trailing digits, wherever possible.

The above procedure doesn't always generate the optimum statistical scanning path because cases may arise, for example, in which $H(k_1) + H(k_2|k_1) > H(k_1') + H(k_2'|k_1')$ while $H(k_1') > H(k_1)$. Blokh claims, however, that in almost all cases a minimum or near-minimum description is obtainable by this method.

It will be recognized that Blokh's formulation of the minimum representation problem is similar to the optimum statistical coding problem of information theory. A basic theorem of Shannon (23) yields the following relationship when applied to the problem at hand: $\bar{n} \geq \bar{n}_o \geq H_o / \log_2 n$, where $\bar{n}_o$ is the average word length of the optimum statistical code, $\bar{n}$ is the average word length of an arbitrary code, and $H_o = -\sum_{i=1}^{P} P_i \log_2 P_i$ is the uncertainty[*] (entropy, information) of the system of patterns, expressed in bits. It is worth observing that a necessary condition for the achievement of the optimum statistical code by Blokh's scheme is that each cell chosen have the maximum possible conditional uncertainty, i.e., each cell must divide the patterns into m equally probable groups, depending upon its color. For the case where $m = 2$, and the patterns are equiprobable, this condition reduces to the familiar "binary division" property discussed previously, and Shannon's formula reduces to the familiar condition $\bar{n}_0 \geq \{\log_2 P\}$. Hence Blokh's technique may be regarded as an extension of the previous techniques to the case of unequally probable patterns.

A second serious shortcoming of the method is its lack of provision for dealing with distortion and noise. It is still possible to compute Blokh's cell significance criterion, H(c), when noise is present. However, it will be shown in Section 1.6 that this criterion is no longer valid in the noisy case. This limits the applicability of the Blokh technique in practice to those relatively rare situations in which the noise level is insignificant.

In Section 1.6 of this report and Chapter III of UCLA Report 62-68, an alternate procedure for finding significant cells when noise is present will be given.

[*] The various names given for the function H_0 are all in common use. For uniformity, the term <u>uncertainty</u> will be used throughout this report.

1.6 The Conditional Uncertainty Criterion for Cell Significance

In this section, a measure of cell significance will be introduced and justified on the basis of intuitive ideas concerning the notion of "significance". A few such ideas have been discussed in the introduction, where qualitative observations were made concerning the effects of noise, variations of a priori probabilities, etc. upon the significance of cells. In Section 1.6.2 of this chapter, these qualitative observations will be made more precise. Where possible, it will be shown analytically that the significance measure introduced here conforms to the intuitive requirements. The final justification for the significance measure is experimental and is described in Chapters III and IV of UCLA Report 62-68.

1.6.1 The Conditional Uncertainty

Let us suppose, as in Section 1.5.4, that we are dealing with a set of characters $N_1, N_2, \ldots, N_K$ having a priori probabilities of occurrence $p(N_1), p(N_2), \ldots, p(N_K)$ respectively, and that the characters are in the form of binary digit matrices containing IxJ cells c_{ij} (Fig. 1.19). Let $H(N) = -\sum_{k=1}^{K} p(N_k) \log_2 p(N_k)$ denote the uncertainty (measured in bits) in the system N consisting of the events $N_1, N_2, \ldots, N_K$ (the "event N_k" signifies, of course, the occurrence of character N_k). On the basis of a large number of representative samples of each character, it is possible to compute (e.g., by using Baran's Recognition technique*) the a posteriori probabilities $p(N_k/w_{ij})$** and $p(N_k/b_{ij})$** for each cell C_{ij} and each character N_k. It is therefore also possible to compute the conditional uncertainty in the system N, given that C_{ij} has been observed:

$$H(N/C_{ij}) = -\left[\sum_{k=1}^{K} p(N_k, w_{ij}) \log_2 p(N_k|w_{ij}) + \sum_{k=1}^{K} p(N_k, l_{ij}) \log_2 p(N_k|l_{ij})\right) \quad \text{II-1}$$

The smaller the value of $H(N/C_{ij})$, the smaller the average uncertainty concerning the identity of the unknown character is after C_{ij} has been observed. It is, therefore, natural to make the following claim: the smaller $H(N/C_{ij})$ is, the greater the significance of C_{ij} is.

Using similar statistical techniques, the process can be continued to determine the significance of a group of q cells. We have:

$$H(N|C_{m1}, C_{m2}, \cdots, C_{mq}) = -\sum_{k=1}^{K}\left[\sum_{\tilde{V}} p(N_k, v_{m1}, v_{m2}, \cdots, v_{mq}) \log_2 p(N_k|v_{m1}, v_{m2}, \cdots, v_{mq})\right],$$

where the variables v_{mi} take on the values l_{mi} and w_{mi}, and the sum in brackets is taken over all possible resulting values of the vector $\tilde{V} = (v_{m1}, \cdots, v_{mq})$. Again, it is asserted that the smaller $H(N/C_{m1}, C_{m2}, \ldots, C_{mq})$ is, the greater the significance of the group of cells $C_{m1}, C_{m2}, \ldots, C_{mq}$ is. The probabilities $P(N_k, \tilde{V})$ and $P(N_k/\tilde{V})$ can be satisfactorily approximated by formulas such as (I-1), provided that the cells $C_{m1}, C_{m2}, \ldots, C_{mq}$ are sufficiently independent. The conditional uncertainty, given the information contained in a group of q cells, will be referred to as a <u>qth-order conditional uncertainty</u>.

* Section 1.5.4.

** The notation introduced in Section 1.5.4 is adhered to in this section.

It is clear that the computational difficulties increase very rapidly with increasing q. In fact, there are $Kx2^q$ terms in the expression for the qth order conditional uncertainty. Moreover, the individual terms become increasingly more difficult to compute as q increases. For this reason, it is not practical to compute conditional uncertainties for orders higher than the first few. An alternate approximate procedure for finding significant groups of cells, without incurring the computational difficulties of higher order conditional uncertainties, will be presented in Section 1.6.3.

It is evident that Blokh's criterion H(c) for finding significant cells is similar in some respects to the conditional uncertainty criterion H(N/c) introduced here. It will now be shown that the two criteria are equivalent <u>in the absence of noise</u>, but that in the noisy case only the criterion given here is reliable. Consider the following basic relationship between joint and conditional uncertainties:*

$$H(c,N) = H(N|c) + H(c) = H(N) + H(c|N)$$

In the noiseless case, H(c/N)=0 since the color of any cell c is uniquely determined once the identity of the character is given. Hence, for ideal characters, H(c,N)=H(N/c)+H(c)=H(N)= constant. Clearly, the larger H(c) is, the smaller H(N/c) must be, and vice versa. Blokh associated cell significance with <u>large</u> values of H(c), whereas <u>small</u> values of H(N/c) are used in this study as a criterion of cell significance. Hence, the two criteria are indeed equivalent when noise is absent.

* See Feinstein (7), pp. 21

When noise is present, on the other hand, large values of H(c) may be due to the <u>noise</u> rather than to actual significance of the cell c. Under these conditions the conditional uncertainty criterion H(N/c) is a far more reliable measure of cell significance, for it measures directly the pertinent quantity, i.e. the uncertainty in the identity of the unknown pattern. To illustrate this point, suppose that cell c is ideally white in each of the possible characters. Let us now assume the following noise distribution: for each character, there is a probability of 1/2 that cell c will appear <u>black</u> rather than white. Since this type of noise is independent of the character to be recognized, cell c still yields no information concerning the identity of the unknown character, and hence should be given the lowest possible significance rating. However, H(c) = 1 bit since there is an equal probability of c being white or black. But this is the <u>maximum</u> possible value for H(c) in a two-tone pattern, and hence, according to Blokh's criterion, cell c would be falsely considered maximally significant. On the other hand, it is readily computed that with the noise present $p(N_i|b) = p(N_i|w) = p(N_i)$ and $p(N_i, b) = p(N_i, w) = \frac{p(N_i)}{2}$, whence $H(N/c) = -\left(\sum_{k=1}^{K} p(N_k, b)\log_2 p(N_k) + \sum_{k=1}^{K} p(N_k, w)\log_2 p(N_k)\right)$
$= -\left(\sum_{k=1}^{K}\left[\frac{p(N_k)}{2}\log_2 p(N_k) + \frac{p(N_k)}{2}\log_2 p(N_k)\right]\right) = H(N)$. But H(N) is the maximum possible value which H(N/c) can assume (see next section). Therefore, according to the H(N/c) criterion, cell c would be given the <u>lowest</u> possible significance ranking, as it should be. This rather extreme example tends to support the conjecture that H(N/c), but not H(c), is a reliable measure of

cell significance in the presence of noise. More will be said about this in the next section.

1.6.2 Justification of the Conditional Uncertainty Criterion

In this section it will be shown that the conditional uncertainty function has certain properties which tend to justify its use as a measure of cell significance. First, however, it is convenient to introduce the following notation:

(1) $S(c) = H(N) - H(N/c)$

(2) $S(c_i/c_j) = H(N/c_j) - H(N/c_i, c_j)$

(3) $S(c_{n1}, c_{n2}, \ldots, c_{nq}) = H(N) - H(N/c_{n1}, \ldots, c_{nq})$

where H(N), H(N/c), etc. are as defined in the preceding section. The function S(c) will be regarded as a numerical measure of the significance of the cell c. Similar remarks hold for the functions $S(c_i/c_j)$, $S(c_{n1}, c_{n2}, \ldots, c_{nq})$, etc. Clearly, the above definitions are in accordance with the previous assertion that cell significance is a decreasing function of the appropriate conditional uncertainty. Moreover, the function S(c) possesses the following property which one would naturally require of a valid measure of significance:

Property A. $S(c) \geq 0$, with equality if and only if $p(N_k/w_c) = p(N_k/b_c) = p(N_k)$ for all k.*

* This condition for S(c) = 0 is of course precisely the circumstance under which one would say "cell c yields no information concerning the identity of the unknown character". In other words, S(c) = 0 corresponds to the case of zero cell significance, as it should.

<u>Proof</u>: This property is an immediate consequence of Shannon's Fundamental Inequality which states that $H(N/c) \leq H(N)$ when particularized to the case considered here. Furthermore, it is shown in Feinstein (7), pages (15-16), that the condition for equality is as stated above.

The following property of the significance function is a special case of the effects of new information concerning the unknown character upon previously estimated cell significance ratings.

<u>Property B</u>. The quantity $S(c_i/c_j)$ denotes the significance to be attributed to c_i, on the assumption that c_j has been observed. If the cells c_i and c_j are highly correlated, i. e. if the state of cell c_i can be predicted with great certainty if that of c_j is known, and vice versa, we should have $S(c_i/c_j) \cong o$ and $S(c_j/c_i) \cong o$. In particular, $S(c/c) = o$ for any cell c.

<u>Proof</u>: It is sufficient to show that $H(N/c_j) \cong H(N/c_i, c_j)$. Assume for definiteness that c_i is almost always white when c_j is white, and vice versa. Then, $p(N_k, w_i) = p(N_k, b_j, w_i) + p(N_k, w_j, w_i) \cong 0 + p(N_k, w_j, w_i) \cong p(N_k, w_j)$. Also,

$$p(N_k|w_i) = \frac{p(N_k, w_i)}{p(w_i)} \cong \frac{p(N_k, w_i, w_j)}{p(w_i, w_j)/p(w_j|w_i)} \cong \frac{p(N_k, w_i, w_j)}{p(w_i, w_j)} \cong \frac{p(N_k, w_j)}{p(w_j)}$$

$$\cong p(N_k|w_i, w_j) \cong p(N_k|w_j).$$

Similar relations hold if "w" is replaced by "b" in the above formulas.

Hence $H(N|c_j) = -\left[\sum_{k=1}^{K} p(N_k, w_j) \log_2 p(N_k|w_j) + \sum_{k=1}^{K} p(N_k, b_j) \log_2 p(N_k|b_j)\right]$

$$\cong -\left[\sum_{k=1}^{K} p(N_k, w_j, w_i) \log_2 p(N_k|w_j, w_i) + \sum_{k=1}^{K} p(N_k, b_j, b_i) \log_2 p(N_k|b_j, b_i)\right] = H(N|c_i, c_j),$$

completing the proof.

The dependence of the cell significance function upon the a priori probabilities has also been discussed in the introduction. The following property is stated as an example of the dependence.

Property C. Suppose that cell c is always black in N_1 and always white in the remaining characters. In other words, the significance of c depends upon the fact that this cell differentiates between N_1 and the remaining characters. Similarly, suppose that cell c_2 is always black in N_2, always white in the remaining characters, and that $1/2 > p(N_1) > p(N_2)$. Then $S(c_1) > S(c_2)$.

Proof: It is necessary to show that $H(N/c_1) < H(N/c_2)$ under the stated conditions. We begin with the relations*

$$H(N, c_1) = H(N) + H(c_1/N) = H(N/c_1) + H(c_1)$$

$$H(N, c_2) = H(N) + H(c_2/N) = H(N/c_2) + H(c_2)$$

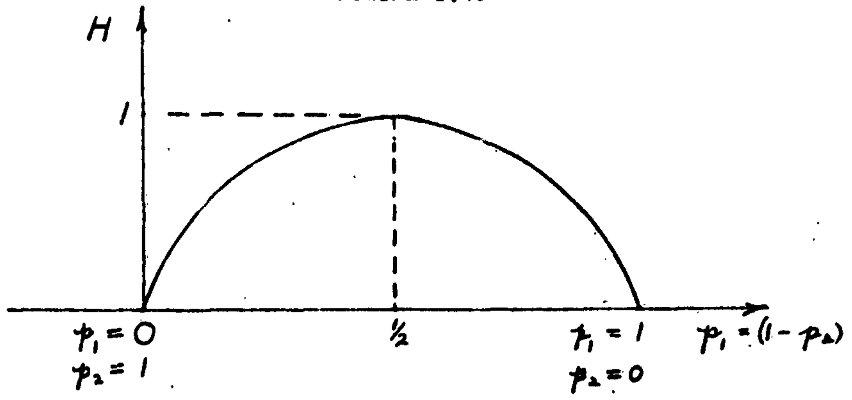
Under the conditions of the hypothesis, it is clear that $H(c_1/N) = H(c_2/N) = 0$. Therefore, we have $H(N) = H(N/c_1) + H(c_1) = H(N/c_2) + H(c_2)$. The proof now reduces to showing that $H(c_2) < H(c_1)$. This is quite simple: $H(c_1)$ is the uncertainty of the 2-event system whose outcomes are "c black" or "c white". The probabilities of these events are $p(b_1) = p(N_1)$ and $p(W_1) = p(N_2) + p(N_3) + \dots + p(N_K) = 1-p(N_1)$ respectively. Similar statements apply to $H(c_2)$. Referring to the 2-event uncertainty curve shown in Figure 1.40, it is seen that the uncertainty function is monotonically increasing for $0 \leq p_1 \leq 1/2$. Since $1/2 > p(N_1) > p(N_2)$ by hypothesis, it can, indeed, be

* Khinchin (16), p. 6.

concluded that $H(c_2) < H(c_1)$, completing the proof.

Property C, as stated above, can be generalized considerably. First of all, the condition that c_1 be black in N_1 (rather than white) and white (rather than black) in $N_2, N_3, \ldots, N_K$ is clearly arbitrary. The same statement applies to the color of c_2. Moreover, the hypothesis that $p(N_2) < p(N_1) \leq 1/2$ can be replaced by the more general hypothesis that $\max\{p(N_1), 1-p(N_1)\} < \max\{p(N_2), 1-p(N_2)\}$, since this is a sufficient condition for $H(c_1) > H(c_2)$.

FIGURE 1.40



The following property is far stronger than those so far considered. Any significance function satisfying it must be immediately accepted as valid. Unfortunately, it is not possible to prove that the significance function considered here has this property except in a rather restricted sense.

<u>Property D:</u> If $S(c_1) > S(c_2)$, then the average error rate in the identification of the unknown character will be lower if identification is based on the set of probabilities $p(N_k/c_1)$ then if it is based on the set of probabilities $p(N_k/c_2)$. A similar statement applies if $S(c_{n1}, c_{n2}, \ldots) > S(c_{m1}, c_{m2}, \ldots)$.

Remarks: In the Baran Recognition scheme, the unknown character is always identified as the character having the greatest a posteriori probability. This maximum a posteriori probability is, therefore, approximately equal to the probability that the unknown character is, indeed, the character that it is identified as. In other words, the average correct identification rate is approximately equal to the average value of the maximum a posteriori probability. In view of this, and recalling the definition of S(c), Property D can be restated as follows:

Let C(H) denote the class of complete K-event systems $\{p_1, p_2, \ldots, p_K\}$ having the uncertainty H. Suppose that the probabilities $\{p_1, p_2, \ldots, p_K\}$ obey a K-variate probability density function $\psi(p_1, p_2, \ldots, p_K)$. Then, according to property E, if $H_2 > H_1$, then $E(p_{m1}) > E(p_{m2})$, where $E(p_{m1})$ denotes the expected value of the maximum probability of the K-event systems $C(H_1)$, and $E(p_{m2})$ denotes the expected value of the maximum probability of the systems $C(H_2)$. The expected values are, of course, given by the formulas:

$$E(p_{m1}) = \int \cdots \int_{p_1, p_2, \cdots, p_K \in C(H)} \max(p_1, p_2, \cdots, p_K) \frac{\psi(p_1, p_2, \cdots, p_K)}{P(H_1)} \, dp_1 \, dp_2 \cdots dp_K$$

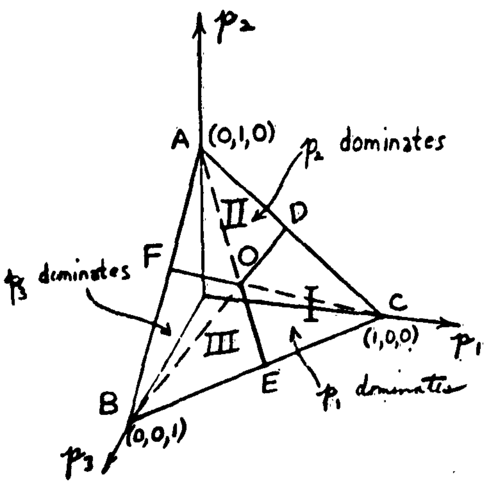
$$E(p_{m2}) = \int \cdots \int_{C(H_2)} \max(p_1, p_2, \cdots, p_K) \frac{\psi(p_1, p_2, \cdots, p_K)}{P(H_2)} \, dp_1 \, dp_2 \cdots dp_K$$

It is easy to see that Property D holds for certain trivial cases, e.g. when K = 2. For, once H is specified for a two-event system, the two probabilities are uniquely determined. From Figure 1.40, however, it is evident that the maximum of the two probabilities is a decreasing function of H. Therefore Property D is satisfied. Property D must also hold if H assumes its maximum possible value, $\log_2 K$, for then $p_{m2} = 1/K = p_1^{(2)} = p_2^{(2)} = \cdots = p_K^{(2)}$ (where p_{mj} denotes the maximum probability possible in a system having uncertainty H_j, and the superscripts j indicate that the probabilities are from a system having the uncertainty H_j). If $H_1 < \log_2 K$, then at least one of the $p_i^{(1)}$'s must be larger than $1/K = p_{m2}$, which proves the assertion. Similarly, Property D holds in case $H_1 = 0$, for then $p_{m1} = 1 > p_i^{(2)}$ $(i=1,2,\cdots,K)$.

For $K > 2$, $H_1 > 0$, and $H_2 < \log_2 K$, the situation is quite complicated since the probabilities $p_i^{(1)}$ and $p_i^{(2)}$ are by no means uniquely determined. For some configurations of the probabilities $p_i^{(1)}$ and $p_i^{(2)}$, it may indeed occur that the maximum p_{m2} of the $p^{(2)}$ exceeds the maximum p_{m1} of the $p_i^{(1)}$ even though $H_1 < H_2$. If property D holds, however, this will not occur "on the average".

In order to gain some insight into the relationships between the uncertainty in a system and the maximum probability, it is worthwhile to consider first the relatively simple case where K = 3. Consider a three-dimensional Cartesian coordinate system with axes labeled p_1, p_2, and p_3 (Fig. 1.41). The various possible probability configurations for a system consisting of three events can be represented by points in three-dimensional space. Clearly, for complete systems these points lie on the plane $p_1 + p_2 + p_3 = 1$. Moreover, since each probability must be positive, the points are restricted to the triangle ABC of Figure 1.41. The lines OD, OF, and OE are segments of the perpendicular bisectors of the sides of triangle ABC. It is evident that

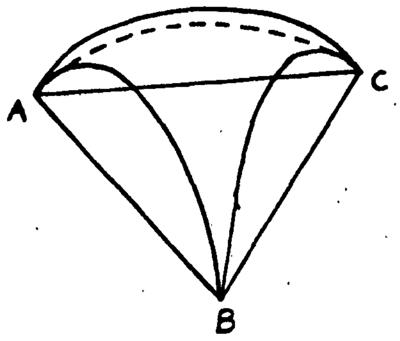
FIGURE 1.41



in Region I, p_1 is the dominant probability (i.e. $p_1 > p_2$ and $p_1 > p_3$); similarly, in Regions II and III, p_2 and p_3 dominate respectively. On the line OE, $p_1 = p_3$. On OD, $p_1 = p_2$, and on OF, $p_2 = p_3$. At the point O, of course, $p_1 = p_2 = p_3 = 1/3$.

For each point on triangle ABC there is an associated uncertainty value $H = -\left[\sum_{i=1}^{3} p_i \log_2 p_i\right]$. A sketch of the corresponding surface is shown in Figure 1.42. The point of maximum uncertainty lies at the center O of the triangle (i.e. the point (1/3, 1/3, 1/3)). The edges AB, BC, and AC of the triangle correspond to points for which $p_1 = 0$, $p_2 = 0$, or $p_3 = 0$, respectively. Accordingly, the conditional uncertainty surface follows the

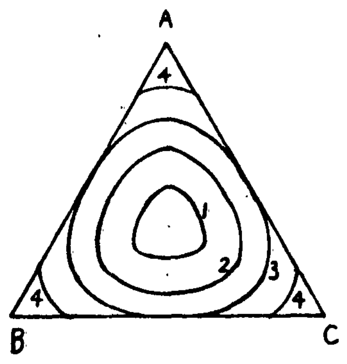
FIGURE 1.42



familiar two-dimensional uncertainty curve (Fig. 1.40) above these lines.

The locus of points on triangle ABC having a given uncertainty value is known as a "level curve" of the uncertainty function. Evidently, the level curve corresponding to a specified value of H, say H_0, is simply the intersection of the uncertainty surface with a plane parallel to ABC, and at a distance $d = H_0$ from ABC. In Figure 1.43, level curves corresponding to various values of H (and d) have been sketched. When $d > \log_2 3$, the plane does not intersect the uncertainty surface, and no curve is formed. This corresponds to the fact that H cannot exceed $\log_2 3$ bits in a system consisting of three events.

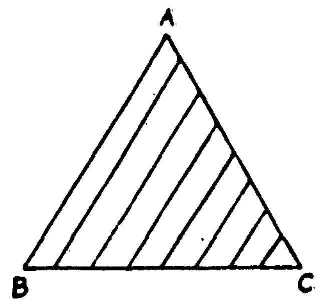
FIGURE 1.43



When $d = \log_2 3$, the level curve consists of the single point O. As d decreases, closed curves such as 1 and 2 are formed. When $d = 1$ ($H = 1$ bit), curve 3 is formed. This curve touches the edges of triangle ABC at their midpoints. This is in agreement with the fact that 1 bit is the maximum possible uncertainty in a two-event system. Level curves corresponding to values of H less than 1 bit consist of three segments (e.g. curve 4). Finally, when $d = H = 0$, the level curve consists of the three points A, B, and C only. This is in accordance with the fact that at these three points, the probability configurations (p_1, p_2, p_3) are (0,1,0), (0,0,1), and (1,0,0) respectively, so that there is indeed no uncertainty.

Level curves of the probabilities p_1, p_2, and p_3 can also be constructed on triangle ABC. In Figure 1.44, level curves of p_1 have been

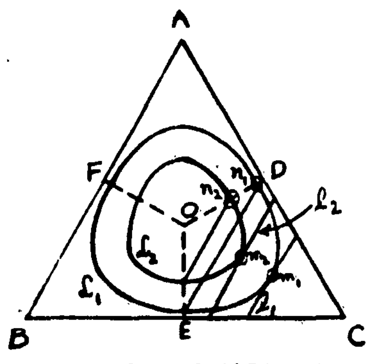
FIGURE 1.44



drawn. These are simply a series of straight lines parallel to AB. The value of p_1 corresponding to a given level curve is proportional to the distance between the level curve and AB. Entirely analogous statements apply to the level curves of p_2 and p_3, which are straight lines parallel to BC and AC respectively.

Property D can now be discussed geometrically. Given the two complete 3-event systems P and Q, with $H(P) < H(Q)$, we begin by constructing the level curves corresponding to $H_1 = H(P)$ and $H_2 = H(Q)$, as in Figure 1.45. According to Property D, if one examines a large sample of probability configurations which are elements of $C(H_1)$, and averages the maximum probabilities, the result will be greater than the corresponding average taken over a large representative sample of elements of $C(H_2)$. Alternately, if Property D holds, the line integral $I_1 = \oint_{\mathcal{L}_1} p_{m_1} \frac{\psi(p_1, p_2, p_3)\, d\mathcal{L}_1}{P(H_1)}$ must exceed the corresponding integral I_2 taken over $\mathcal{L}_2$. There-

FIGURE 1.45



SMALL CIRCLES INDICATE LOCATIONS OF n_1, n_2, m_1, and m_2

fore, in order to determine if Property D holds, it is necessary to evaluate line integrals such as I_1 for various values of H_1. Before these computations can be carried out, however, $\psi(p_1, p_2, p_3)$ must be known. It can be seen that in general this density function will depend upon the a priori probabilities of the characters, the amount of noise in the characters, the way the noise is distributed, and other variables as well. For this reason, it does not seem possible to make any general statements about the validity of Property D. Instead, the following weaker property of the significance function will be proved.

Property D′. Let C(H) again denote the class of complete K-event systems $\alpha = \{p_1^\alpha, p_2^\alpha, \ldots, p_K^\alpha\}$ having uncertainty H, and let $p_{m\alpha}$ denote the maximum of the probabilities $\{p_1^\alpha, p_2^\alpha, \ldots, p_K^\alpha\}$. X(H) will denote the maximum of the probabilities $\{p_{m\alpha}\}$, where α runs through C(H). Assume that $H_1 < H_2$. Then $X(H_1) > X(H_2)$.

Preliminary Remarks: It is readily verified from Figure 1.45 that Property D holds in the 3-dimensional case. The reader will recall that the curves $\mathcal{L}_1$ and $\mathcal{L}_2$ represent level functions of H corresponding to the values $H = H_1$ and $H = H_2$ respectively, where $H_2 > H_1$. Restricting attention to Region I of the figure, we observe that the point m_1 corresponds to the probability configuration in which p_1 assumes its maximum possible value consistent with the restriction that the uncertainty of the system be H_1. This must be so, since the level curve ℓ_1 (of p_1) which passes through m_1 is further from

AB than any other level curve of p_1 which meets $\mathcal{L}_1$. By symmetry, the value of p_1 at m_1 is identical with the maximum values attainable by p_2 and p_3 subject to the restriction that the uncertainty of the system be H_1 . Therefore, this value of p_1 is equal to $X(H_1)$. Analogous statements apply to the point m_2 on $\mathcal{L}_2$. Since m_2 is closer to AB than m_1 is, it follows that $X(H_2) < X(H_1)$, as it should be. The same argument can be applied to any two distinct level curves; thus, Property D' is satisfied in the 3-dimensional case.

As an adjunct to Property D', the following statement concerning the minimum N(H) of the $p_{m\alpha}$, subject to the constraint that $\alpha \in C(H)$, can be made.

Property D'' If $H_1 < H_2$, then $N(H_1) > N(H_2)$.

Unfortunately, the author has not succeeded in proving this in general. Referring to Figure 1.45 , we observe that n_1 and n_2 are the points corresponding to the probability configurations in which $p_3 = N(H_1)$ and $p_3 = N(H_2)$, respectively. Obviously, n_1 corresponds to a larger value of p_2 than n_2 does, as required by Property D''. Applying the same argument to any two distinct level curves, we conclude that Property D'' is true in the 3-dimensional case. Incidentally, it can be seen that Property D'' is a consequence of Property D' in the 3-dimensional case, as follows. From Figure 1.45 , it is evident that if Property D' is satisfied by two distinct level curves, but Property D'' were not, the two level curves would have to intersect somewhere. But that is impossible since the uncertainty value of any point on triangle ABC is unique. Hence Property D' implies Property D'' in the 3-dimensional case. Perhaps this argument can be extended to higher dimensions as well. Then, in view of the general proof of Property D' given below, we would have a general proof for Property D''.

<u>Proof of Property D'.</u> There exists some $\alpha' \in C(H)$ such that $X(H) = \max\{p_1^{\alpha'}, p_2^{\alpha'}, \ldots, p_K^{\alpha'}\}$. Suppose for definiteness that $X(H) = p_1^{\alpha'}$. The first step of the proof consists in showing that $p_2^{\alpha'} = p_3^{\alpha'} = \ldots = p_K^{\alpha'} = \frac{1-p_1^{\alpha'}}{K-1}$. Holding H constant, we now wish to find the values of $p_2^{\alpha}, p_3^{\alpha}, \ldots, p_K^{\alpha}$ for which p_1^{α} is maximum. For this purpose, Lagrange's Method of Multipliers will be employed. We have the two equations

(1) $p_1^{\alpha} = 1 - \sum_{k=2}^{K} p_k^{\alpha}$ (to be maximized)

(2) $\left(1-\sum_{k=2}^{K} p_k^{\alpha}\right)\log_2\left(1-\sum_{k=2}^{K} p_k^{\alpha}\right) + \sum_{k=2}^{K} p_k \log_2 p_k = \text{constant}$ (constraint equation).

Multiplying (2) by λ, differentiating the result, and adding this to the differential of (1) yields: $\lambda\left[\sum_{k=2}^{K}\left(\log_2 p_k^{\alpha} - \log_2\left(1-\sum_{i=2}^{K} p_i^{\alpha}\right) dp_k\right] - \sum_{k=2}^{K} dp_k^{\alpha} = 0.$

Hence $\lambda = \frac{1}{\log_2 p_k^{\alpha} - \log_2\left(1-\sum_{i=2}^{K} p_i^{\alpha}\right)}$ for $k = 2, \cdots, K$. But this implies that $p_2^{\alpha'} = p_3^{\alpha'} = \cdots = p_K^{\alpha'} = \frac{1-p_1^{\alpha'}}{k-1}$ which is what we started out to prove.

Now we can prove Property D', as follows: We can regard a K-event system $\alpha = \{p_1^{\alpha}, p_2^{\alpha}, \ldots, p_K^{\alpha}\} \in C(H)$ as a two-event system β having the possible outcomes "outcome #1" (with probability p_1^{α}), or "not outcome #1" (with probability $(1-p_1^{\alpha})$), followed by the (K-1)-event system $\gamma = \left\{\frac{p_2^{\alpha}}{1-p_1^{\alpha}}, \ldots, \frac{p_K^{\alpha}}{1-p_1^{\alpha}}\right\}$. It is easy to show that $H = -\left[p_1^{\alpha}\log_2 p_1^{\alpha} + (1-p_1^{\alpha})\log_2 (1-p_1^{\alpha})\right.$

$+ (1-p_1^{\alpha})H_{\gamma} = H_{\beta} + (1-p_1^{\alpha})H_{\gamma}$,* where H_{β} and H_{γ} denote the uncertainties of the systems β and γ respectively. Therefore, for the system α', we can write:

$$H = -\Big[p_1^{\alpha'} \log_2 p_1^{\alpha'} + (1-p_1^{\alpha'})\log_2(1-p_1^{\alpha'}) + (1-p_1^{\alpha'})\Big(\sum_{k=2}^{K}\Big(\frac{1}{k-1}\Big)\log_2 \frac{1}{k-1}\Big)\Big] = -\Big[p_1^{\alpha'} \log_2 p_1^{\alpha'} + (1-p_1^{\alpha'})\log_2(1-p_1^{\alpha'}) + (1-p_1^{\alpha'})\log_2\Big(\frac{1}{k-1}\Big)\Big].$$

Now, considering H as a function of $p_1^{\alpha'}$ and differentiating, we find:

$\frac{dH}{dp_1^{\alpha'}} = -\Big[\log_2 p_1^{\alpha'} + 1 - \log_2(1-p_1^{\alpha'}) - 1 - \log_2\Big(\frac{1}{k-1}\Big)\Big] = \log_2\Big[\frac{1-p_1^{\alpha'}}{(k-1)p_1^{\alpha'}}\Big]$. Clearly, $\frac{dH}{dp_1^{\alpha'}} = 0$ when $\Big[\frac{1-p_1^{\alpha'}}{(k-1)p_1^{\alpha'}}\Big] = 1$, hence $(k-1)p_1^{\alpha'} = 1-p_1^{\alpha'}$, or $p_1^{\alpha'} = 1/k$. If $p_1^{\alpha'} > 1/k$, it is readily verified that $\frac{1-p_1^{\alpha'}}{(k-1)p_1^{\alpha'}} < 1$. Since $p_1^{\alpha'} \geq \frac{1}{k}$ always, it follows that $\frac{dH}{dp_1^{\alpha'}} \leq 0$ always, i.e. H is a strictly decreasing function of $p_1^{\alpha'} = X(H)$ for $\frac{1}{k} \leq p_1^{\alpha'} \leq 1$. This completes the proof of Property D$'$.

1.6.3 Correlations Between Cells.

In the first section of this chapter the difficulty of computing higher-order conditional uncertainties was pointed out. Therefore, in order to test the conditional uncertainty criterion experimentally, it is necessary to make use of an alternate procedure for finding significant sequences of cells which is based upon Property B of the significance function.**By means of this procedure, it is possible to find highly significant sequences of cells without having to compute higher-order conditional uncertainties. The pro-

* See Feinstein (7), p. 5, Lemma 3.
** Refer to the preceding section.

cedure is as follows:

(1) The first-order conditional uncertainty $H(N/C_i)$ is computed for each cell C_i in the character. The cell C_s having the smallest conditional uncertainty value is noted (this cell, it will be recalled, is the most significant one).* A certain number of cells having very high conditional uncertainties can be eliminated from consideration immediately.

(2) The correlation coefficient $\rho_{is} = \frac{\sigma_{is}}{\sigma_i \sigma_s}$ between C_s and each cell C_i not yet eliminated is computed according to the formulas

$$\sigma_{is} = E[(i-E(i))(s-E(s))], \quad \sigma_i = E[(i-E(i))^2], \quad \sigma_s = E[(s-E(s))^2],$$

where E(x) denotes the average value of the variable x based upon a large number of representative samples of all the characters (properly weighted according to the assumed a priori probabilities); i is a variable whose value is +1 when C_i is black, -1 otherwise; s = +1 or -1 according as C_s is black or white. The Cell C_s with respect to which the correlations are taken will be called the "pivot" cell.

It is a well known fact that $-1 < \rho_{is} < +1$. If $\rho_{is} = \pm 1$, that is, if C_i and C_s are "highly correlated", then $i = \pm s$ almost always. Hence, C_i and C_s yield essentially the same information. According to Property B, therefore, $s(C_i/C_s) \cong 0$ for any cell C_i which is highly correlated

* In case of a tie, an arbitrary choice should be made.

with C_s . Hence, if $|\rho_{is}| \cong 1$, it can be concluded immediately that cell C_i is rendered insignificant by observation of C_s . Any sequence of significant cells beginning with C_s would therefore not contain cell C_i . All such cells C_i can therefore be eliminated from further consideration. All cells surviving the second sieving must also have survived the first sieving process, which was on the basis of conditional uncertainty. Therefore, these cells can be regarded as both significant and independent of C_s . If the first two sievings have been sufficiently severe, that is, if enough cells have been eliminated each time, the number of remaining cells will be small and an acceptable reduced scanning path will have been achieved. If it is desired to eliminate more cells, a second sieving process can be carried out by selecting a second "pivot" cell C_s' from among those not yet eliminated, computing correlation coefficients $\rho_{is'}$ between it and all remaining cells, and eliminating all cells for which $|\rho_{is'}| \cong 1$. The pivot cell C_s' can be selected by one of the following simple methods, each of which require no further computation:

(a) Choose as C_s' that cell, distinct from C_s , having the smallest conditional uncertainty value;*

(b) Choose as C_s' that cell, distinct from C_s , having the smallest value of $|\rho_{is}|$.*

* In case of ties, an arbitrary choice can be made.

The process can be repeated as many times as necessary. The final result is a minimal or near-minimal scanning path. In Chapters III and IV of Report 62-68, computer programs for carrying out the above process are described, and the results of actual trials are discussed.

In applying the technique described above, the following difficult question must be answered: How many cells should be eliminated at each stage? Obviously, the answer to this question depends upon the nature of the character samples dealt with. As a general rule, however, it is advisable to retain a fairly large number of cells after the first (conditional uncertainty) sieving, since most of the cells obtained will be redundant. Subsequent correlation sieving will, therefore, result in elimination of a large number of cells, leaving only a few independent, significant ones. If too many cells are eliminated in the first sieving, there is a possibility that an insufficient number of independent ones will remain and satisfactory recognition will hence not be achieved.

It is worthwhile to consider the computational simplicity of the correlation-sieving process in comparison to the higher-order conditional uncertainty sieve. Whereas the conditional uncertainty sieving becomes more difficult by an order of magnitude at each stage, the correlation process actually becomes <u>simpler</u> since there are fewer cells to be dealt with at each stage.

BIBLIOGRAPHY

1. Baran. Paul
"A Probability Matrix Pattern Comparator"
M.S. Thesis, Department of Engineering,
University of California, Los Angeles, May 1959.

2. Blokh, E.L.
"Concerning the Minimal Representation Problem"
Radio Engineering, 15: No. 2, 15-24, February 1960

3. Breuer, Melvin Allen
"The Use of a Digital Computer for Studies of Character Recognition"
M.S. Thesis, Department of Engineering, University of California, Los Angeles, September 1960.

4. Bushor, W. E.
"The Perceptron - An experiment in Learning"
Electronics, 33: 56-9, July 22, 1960.

5. Dickinson, W. E.
"A Character Recognition Study"
IBM Journal of Research and Development, 4: 335-48, July 1960.

6. Doyle, Worthie
"Recognition of Sloppy Hand-written Characters", Proceedings of the Western Joint Computer Conference, 17: 133-142, 1960.

7. Feinstein, Amiel
Foundations of Information Theory
McGraw-Hill, New York, 1958.

8. Flores, I. and L. Grey
"Optimization of Reference Signals for Character Recognition Systems"
I. R. E. Transactions on Electric Computers, EC-9: 54-61, March 1961.

9. Gill, A.
"Minimum-Scan Pattern Recognition"
I. R. E. Transactions on Information Theory, IT-5: 52-7, June 1957.

10. Glovazky, A.
"Author's Comment",
I. R. E. Transactions on Information Theory, IT-3: 167, June 1957.

11. Glovazky, A.
"Determination of Redundancies in a Set of Patterns",
I. R. E. Transactions on Information Theory, IT-2: 151-3,
December 1956.

12. Hamming, R. W.
"Error Detecting and Error Correcting Codes"
Bell System Technical Journal, 29: 147-160, April 1950.

13. Harmon, L. D.
"A Line Drawing Pattern Recognizer"
Proceedings of the Western Joint Computer Conference, 17:
351-64, 1960.

14. Horwitz, L. P., and Shelton, G. L.
"Pattern Recognition Using Autocorrelation"
Proceedings of the I. R. E., Special Issue on Computers,
49: 175-184, January 1961.

15. Kharkevich, A. A.
"Principles of Construction for Reading Machines"
Radio Engineering, 15: No. 2, 1-14, February 1960.

16. Khinchin, A. I.
Mathematical Foundations of Information Theory
Translated by R. A. Silverman and M. D. Friedman
Dover Publications, New York, 1957.

17. Lowenschuss, O.
"A Comment on Pattern Redundancy"
I. R. E. Transactions on Information Theory, IT-4: 127,
September 1958.

18. Manelis, Richard
"Character Generation: An Aid for the Testing of Character
Recognition Schemes"
M. S. Thesis, Department of Engineering, University of
California, Los Angeles, June 1962.

19. Minsky, Marvin
"Steps Toward Artificial Intelligence"
Proceedings of the I. R. E., Special Issue on Computers,
49: 8-30, January 1961.

20. Paull, Marvin C.
"A Comment on Pattern Redundancy"
I. R. E. Transactions on Information Theory, IT-5: 34-5, March 1958.

21. Rosenblatt, F.
"The Perception - A Theory of Statistical Separability in Cognitive Systems", Cornell Aeronautical Laboratories, Buffalo, New York, Report No. VG-1196-G-1, January 1958.

22. Selfridge, O.G.
"Pattern Recognition and Modern Computers"
Proceedings of the Western Joint Computer Conference, 91-93, 1955.

23. Shannon, C.E. and Weaver, W.
The Mathematical Theory of Communication
Urbana, University of Illinois Press, 1945.

CHAPTER II - THIN FILM STUDIES

The research outlined in this chapter is described in detail in Technical Report No. 62-37 dated August 1962.

A. STATIC MAGNETIC BEHAVIOR OF MAGNETOSTRICTIVE THIN FILMS

Experimental Studies

a. Stress effects on hysteresis loops
b. Stress effects on critical curve
c. Stress effects on domain wall configurations (Bitter patterns)
d. Film magnetostriction measurements

B. DYNAMIC BEHAVIOR OF MAGNETOSTRICTIVE THIN FILMS

1. Theoretical Model - Equation of Landau and Lifshitz

a. Preliminary considerations, assumptions
b. Solution by analog computer, stress effects on switching
c. Solution by digital computer, stress effects on switching
d. Comparison between analog and digital computer results
e. Computational flow charts for integration and plotting of film dynamic response
f. Effects of parameters $\alpha, \beta, \delta, \phi_o, \lambda$, on film switching response

1. Film Switching Response

   non-destructive pulsing
   destructive pulsing

2. Inverse Peaking Time vs. Switching Field

3. Ferromagnetic Resonant Frequency vs. Switching Field

4. Dynamic Switching Threshold vs. Perpendicular Field

2. Instrumentation for the Study of Stress Effects on Thin Film Switching

a. Description of instrumentation
b. Operation and design considerations

1. Strip Line Design

2. Estimation of Strip Line Available Magnetic Field

3. Estimation of Film Pickup Voltage

4. Mercury Relays, Coaxial Cables, Connectors

C. PIEZOELECTRIC CRYSTALS AND PIEZOELECTRIC MAGNETO-STRICTIVE COUPLING

1. Characteristics of Piezoelectric and Electrostrictive Crystals. Table

2. <u>Theoretical Models of Piezoelectric Behavior</u>

a. General methods of analysis, piezoelectric equations of state, equation of motion, magnitude of piezoelectric constant d.
b. Electrostrictive bar, longitudinal mode (DC strains). See technical report No. 60-82 dated Nov. 1960, pp. IV-9, 10.
c. Electrostrictive bar, thickness mode (DC and transient strains). See technical report No. 60-82 dated Nov. 1960, pp. IV-9, 10 and IV-33, 50 (Appendix B).
d. Electrostrictive bar, bending mode, bimorph (DC strains).
e. Electrostrictive disc, radial mode, (DC strains).
f. Electrostrictive thin wall closed cylinder, radial mode (DC strains).
g. Electrostrictive thin walled split cylinder, flexural mode, (DC strains).
h. Piezoelectric ADP plate X-cut, quartz plate Y-cut (DC strains).

3. <u>Experimental Studies on Electrostrictive Ceramics (DC strains)</u>

a. Electrostrictive disc
b. Electrostrictive cylinder
c. Electrostrictive split cylinder
d. Experimental determination of coupling coefficient.

4. <u>Piezoelectric (electrostrictive) - Magnetostrictive Coupling.</u>

a. Theoretical model of static behavior. Piezoelectric uniaxial and isotropic stress.
b. Experimental studies. Isotropic stress.

APPENDICES

<u>Computer Programs and Experiments</u>

1. Program for computation of hysteresis loops.
2. Program for computation and plotting of static critical switching curves.
3. Description of substrate bending jig for bitter pattern studies.
4. Description of device for bending thin substrates in B-H loop tester.
5. Program for solution of Landau-Lifshitz equation and computer curve plotting.
6. Program for computation and plotting of "inverse peaking time" vs switching field amplitude.
7. Program for computation and plotting of ferromagnetic oscillation frequency vs. switching field amplitude.
8. Program for computation and plotting of dynamic critical switching curve.
9. Design and construction of probe to determine the direction of the earth's magnetic field.
10. Construction of system for production of thin films by vacuum evaporation. Construction of auxiliary instrumentation for magnetic thin film preparation.

CHAPTER III - OPTIMIZATION STUDIES

During the period of June 1961 to May 1962, the emphasis was placed on individual optimization problems which can be studied analytically either exactly or by a sequence of approximations which can be studied analytically.

Admittedly, classes of optimization problems that can be discussed this way is limited, however, it is felt that any additional light one could shed by analytical means is of help.

Technical reports and papers written during this period reflect this attitude.

For details, readers are referred to:

Tech. Report 61-62, also to appear in Automatica.
Tech. Report 61-66, also AIEE Transactions,
Applications and Industry
pp. 125-127, July, 1962.
Tech. Report 61-82, also to appear in J. of Math. Analysis and
Applications.

The above reports take problems from the area of control systems optimization.

An optimization study involving a probabilistic study of relative efficiency of various Variable Structure Computers in carrying out Aitken-Neville interpolation procedure, was undertaken during this period and continued to next year.

DISTRIBUTION LIST

COPIES	AGENCY
2	Assistant Sec. of Def. for Res. and Eng. Information Office Library Branch Pentagon Building Washington 25, D.C.
10	Armed Services Technical Information Agency Arlington Hall Station Arlington 12, Virginia
2	Chief of Naval Research Department of the Navy Washington 25, D.C. Attn: Code 437, Information Systems Branch
1	Chief of Naval Operations OP-07T-12 Navy Department Washington 25, D.C.
6	Director, Naval Research Laboratory Technical Information Officer Code 2000 Washington 25, D.C.
10	Commanding Officer, Office of Naval Research Navy #100, Fleet Post Office New York, New York
1	Commanding Officer, O N R Branch Office 346 Broadway New York 13, New York
1	Commanding Officer, O N R Branch Office 495 Summer Street Boston 10, Massachusetts
1	Bureau of Ships Department of the Navy Washington 25, D.C. Attn: Code 607A NTDS
1	Bureau of Naval Weapons Department of the Navy Washington 25, D.C. Attn: RAAV Avionics Division
1	Bureau of Naval Weapons Department of the Navy Washington 25, D.C. Attn: RMWC Missile Weapons Control Div.
1	Bureau of Naval Weapons Department of the Navy Washington 25, D.C. Attn: RUDC ASW Detection & Control Div.
1	Bureau of Ships Department of the Navy Washington 25, D.C. Attn: Communications Branch, Code 686
1	Naval Ordnance Laboratory White Oaks Silver Spring 19, Maryland Attn: Technical Library

COPIES	AGENCY
1	David Taylor Model Basin Washington 7, D.C. Attn: Technical Library
1	Naval Electronics Laboratory San Diego 52, California Attn: Technical Library
1	University of Illinois Control Systems Laboratory Urbana, Illinois Attn: D. Alpert
1	University of Illinois Digital Computer Laboratory Urbana, Illinois Attn: Dr. J.E. Robertson
1	Air Force Cambridge Research Laboratories Laurence C. Hanscom Field Bedford, Massachusetts Attn: Research Library, CRX2-R
1	Technical Information Officer US Army Signal Research & Dev. Lab. Fort Monmouth, New Jersey Attn: Data Equipment Branch
1	National Security Agency Fort George C. Meade, Maryland Attn: R-4, Howard Campaigne
1	US Naval Weapons Laboratory Dahlgren, Virginia Attn: Head Compution Div., G.H. Gleissner
1	National Bureau of Standards Data Processing Systems Division Room 239, Bldg. 10 Attn: A.K. Smilow, Washington 25, D.C.
1	Aberdeen Proving Ground, BRL Aberdeen Proving Ground, Maryland Attn: J.H. Giese, Chief Compution Lab.
1	Commanding Officer O N R Branch Office John Crerar Library Bldg. 86 East Randolph Street Chicago 1, Illinois
1	Commanding Officer O N R Branch Office 1030 E. Green Street Pasadena, California
1	Commanding Officer O N R Branch Office 1000 Geary Street San Francisco 9, California
1	National Bureau of Standards Washington 25, D.C. Attn: Mr. R.D. Elbourn

COPIES	AGENCY
1	Naval Ordnance Laboratory Corona, California Attn: H.H. Weider
1	George Washington University Washington, D.C. Attn: Prof. N. Grisamore
1	Dynamic Analysis and Control Laboratory Massachusetts Institute of Technology Cambridge, Massachusetts Attn: D.W. Baumann
1	New York University Washington Square New York 3, New York Attn: Dr. H. Kallmann
1	Univ. of Michigan Ann Arbor, Michigan Attn: Dept. of Philosophy, Prof. A. W. Burks
1	Census Bureau Washington 25, D.C. Attn: Office of Asst. Director for Statistical Services, Mr. J.L. McPherson
1	University of Maryland Physics Department College Park, Maryland Attn: Professor R.E. Glover
1	Columbia University New York 27, New York Attn: Dept. of Physics, Prof. L. Brillouin
1	Hebrew University Jerusalem, Israel Attn: Prof. Y. Bar-Hillel
1	Massachusetts Institute of Technology Research Laboratory of Electronics Attn: Prof. W. McCulloch
1	University of Illinois Urbana, Illinois Attn: John R. Pasta
1	Naval Research Laboratory Washington 25, D.C. Attn: Security Systems Code 5266, Mr. G. Abraham
1	National Physical Laboratory Teddington, Middlesex England Attn: Dr. A.M. Uttley, Superintendent, Autonomics Division
1	Diamond Ordnance Fuze Laboratory Connecticut Ave. & Van Ness St. Washington 25, D.C. ORDTL-012, E.W. Channel
1	Harvard University Cambridge, Massachusetts Attn: School of Applied Science, Dean Harvey Brook

COPIES	AGENCY
1	Wright Air Development Division Electronic Technology Laboratory Wright-Patterson AFB, Ohio Attn: Lt. Col. L.M. Butsch, Jr. ASRUEB
1	Laboratory for Electronics, Inc. 1079 Commonwealth Ave. Boston 15, Massachusetts Attn: Dr. H. Fuller
1	Stanford Research Institute Computer Laboratory Menlo Park, California Attn: H.D. Crane
1	General Electric Co. Schnectady 5, N.Y. Attn: Library, L.M.E. Dept., Bldg. 28-501
1	The Rand Corp. 1700 Main St. Santa Monica, California Attn: Numerical Analysis Dept. Willis H. Ware
1	General Electric Research Laboratory P.O. Box 1088 Schenectady, New York Attn: Information Studies Section R. L. Shuey, Manager
1	Stanford Research Institute Menlo Park, California Attn: Dr. Charles Rosen Applied Physics Laboratory
1	New York University New York, New York Attn: Dr. J.H. Mulligan, Jr. Chairman of E.E. Dept.
1	Marquardt Aircraft Company 16555 Saticoy Street P.O. Box 2013 - South Annex Van Nuys, California Attn: Dr. Basun Chang, Research Scientist
1	Texas Technological College Lubbock, Texas Attn: Paul G. Griffith Department of Electrical Engineering
1	L. G. Hanscom Field AF-CRL-CRRB Bedford, Mass. Attn: Dr. H.H. Zschirnt
1	Department of the Army Office of the Chief of Research & Development Pentagon, Room 3D442 Washington 25, D.C. Attn: Mr. L.H. Geiger
1	Bell Telephone Laboratories Murray Hill Laboratory Murray Hill, New Jersey Attn: Dr. Edward F. Moore
1	General Electric Research Lab. P.O. Box 1088 Schenectedy, New York Attn: V.L. Newhouse Applied Physics Section

COPIES	AGENCY
1	National Biomedical Research Foundation Inc. 8600 16th St., Suite 310 Silver Spring, Maryland Attn: Dr. R.S. Ledley
1	University of Pennsylvania Moore School of Electrical Engineering 200 South 33rd Street Philadelphia 4, Pennsylvania Attn: Miss Anna Louise Campion
1	Army Research Office OCR & D Department of Army Washington 2, D.C. Attn: Mr. Gregg McClurg
1	Mr. Paul W. Howerton Room 1053 M. Bldg. Code 163 CIA Washington, D.C.
1	University of Pennsylvania Mechanical Languages Projects Moore School of Electrical Engineering Philadelphia 4, Pennsylvania Attn: Dr. Saul Gorn, Director
1	Applied Physics Laboratory Johns Hopkins University 8621 Georgia Avenue Silver Spring, Maryland Attn: Document Library
1	Bureau of Supplies and Accounts, Chief Navy Department Washington, D.C. Attn: Code W3
1	Prof. E. L. Hahn Dept. of Physics University of California Berkeley 4, California
1	Auerbach Electronics Corporation 1634 Arch St. Philadelphia 3, Pa.
1	National Security Agency Fort George G. Meade, Maryland Attn: R. -42, R. Wiggington
1	Federal Aviation Agency Bureau of Research and Development Washington 25, D.C. Attn: RD-375 Mr. Harry Hayman
1	Federal Aviation Agency Bureau of Research & Development Center Atlantic City, New Jersey Attn: Simon Justman
1	Chief, Bureau of Ships Code 671A2 Washington, D.C. Attn: LCDR. E. B. Mahinske, USN
1	Lincoln Laboratory Massachusettes Institute of Technology Lexington 73, Massachusetts Attn: Library

COPIES	AGENCY
1	Dr. Tsute Yang Digital Systems Group Radio Corporation of America Moorestown, New Jersey
1	Professor C. L. Pekeris, Head Department of Applied Mathematics Weizmann Institute of Science Rehovoth, Israel
1	Mr. Julian H. Bigelow Institute for Advanced Study Princeton, New Jersey
1	Mr. Raoul Sajeva Uiale Legioni Romane 22/7 Milano, Italy
1	Electronics Research Laboratory University of California Berkeley 4, California Attn: Director
1	Mr. Gordon Stanley 7685 South Sheridan Ct. Littleton, Colorado Martin, Denver
1	R. Turyn Applied Research Lab. Sylvania El. Pd. Inc. 40 Sylvan Road Waltham 54, Mass.
1	P. Braffort CETIS Euratom C.C.R. Ispra (Varese), Italy
1	Information Processing Directorate Attn: C. J. Shaw System Development Corporation 2500 Colorado Avenue Santa Monica, California
1	Director Courant Inst. of Mathematical Sciences New York University 4 Washington Square New York 3, New York
1	Dr. Alston S. Householder Oak Ridge National Laboratory Oak Ridge, Tennessee
1	Dr. Milton E. Rose Lawrence Radiation Laboratory University of California Berkeley, California
1	Professor Maria G. Mayer University of California San Diego, California
1	Martin Graham Rice University Houston, Texas

Digital Technology